



NATURAL

Natural

User's Guide

Version 3.1.6 for Mainframes

 **SOFTWARE AG**



This document applies to Natural Version 3.1.6 for Mainframes and to all subsequent releases. Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

© June 2002, Software AG
All rights reserved

Software AG and/or all Software AG products are either trademarks or registered trademarks of Software AG. Other products and company names mentioned herein may be the trademarks of their respective owners.

Table of Contents

User's Guide for Mainframes - Overview	1
User's Guide for Mainframes - Overview	1
Fundamentals	2
Fundamentals	2
Components of Natural	2
Invoking Natural	3
Terminating Natural	3
Terminating an Online Session	3
Terminating a Batch Session	3
Using the ENTER Key	4
Online Help System	4
Help on Natural System Messages	4
Navigating within Natural	5
Invoking a Function from a Menu	5
Invoking a Function with a Command	5
Leaving a Function	6
PF Keys	6
Main Natural Menus	7
Main Menu	7
Development Functions	9
Changing the Library	12
Programming Modes	12
Development Environment Settings	13
Maintenance and Transfer Utilities	15
Debugging and Monitoring Utilities	16
Example Libraries	18
Other Products	18
Natural Editors	19
Terminal Commands	19
Asterisk Notation	20
Tutorial - Getting Started with Natural	21
Tutorial - Getting Started with Natural	21
Session 1 - Creating a Program and a Map	21
Step 1	23
Step 2	23
Step 3	24
Step 4	25
Step 5	26
Step 6	29
Step 7	30
Step 8	30
Step 9	32
Step 10	32
Step 11	33
Step 12	34
Step 13	35
Step 14	35
Step 15	35
Step 16	36
Step 17	36
Step 18	37
Step 19	37
Session 2 - Creating a Local Data Area	38

Step 1	39
Step 2	40
Step 3	41
Step 4	42
Step 5	43
Step 6	43
Step 7	44
Step 8	44
Step 9	46
Step 10	46
Session 3 - Creating a Global Data Area	47
Step 1	47
Step 2	47
Step 3	48
Step 4	49
Step 5	50
Step 6	50
Step 7	52
Session 4 - Creating an External Subroutine	53
Step 1	53
Step 2	54
Step 3	54
Step 4	54
Step 5	56
Session 5 - Editing a Map	57
Step 1	58
Step 2	59
Step 3	61
Step 4	61
Step 5	62
Step 6	64
Step 7	65
Step 8	66
Step 9	66
Step 10	67
Step 11	67
Step 12	69
Session 6 - Invoking a Subprogram	70
Step 1	71
Step 2	71
Step 3	71
Step 4	72
Step 5	72
Step 6	72
Step 7	72
Step 8	73
Step 9	73
Step 10	74
Step 11	76
Tutorial - Using the Map Editor	77
Tutorial - Using the Map Editor	77
Components of the Map Editor	78
Invoking the Map Editor	79
Session 1 - Designing a Map, Line and Field Commands	80
Session 2 - Processing Rules	94
Session 3 - Extended Field Editing	100

Session 4 - INPUT USING MAP	106
Session 5 - WRITE USING MAP, Fields from a View	108
Designing User Interfaces - Overview	114
Designing User Interfaces - Overview	114
Screen Design	115
Screen Design	115
Control of Function-Key Lines - Terminal Command %Y	116
Format of Function-Key Lines	116
Positioning of Function-Key Lines	117
Cursor-Sensitivity	119
Control of the Message Line - Terminal Command %M	120
Positioning the Message Line	120
Message Line Protection	121
Message Line Color	121
Assigning Colors to Fields - Terminal Command %=	122
Outlining - Terminal Command %D=B	124
Statistics Line/Infoline - Terminal Command %X	124
Statistics Line	124
Infoline	126
Windows	127
What is a Window?	127
DEFINE WINDOW Statement	129
INPUT WINDOW Statement	131
Standard/Dynamic Layout Maps	135
Dynamic Layout Maps	135
Multilingual User Interfaces	136
Language Codes	136
Defining the Language of a Natural Object	137
Defining the User Language	138
Referencing Multilingual Objects	139
Programs	140
Error Messages	140
Edit Masks for Date and Time Fields	140
Skill-Sensitive User Interfaces	141
Dialog Design	143
Dialog Design	143
Field-Sensitive Processing	143
*CURS-FIELD and POS(<i>field-name</i>)	143
Simplifying Programming	146
System Function POS	146
Line-Sensitive Processing	147
System Variable *CURS-LINE	147
Column-Sensitive Processing	148
System Variable *CURS-COL	148
Processing Based on Function Keys	149
System Variable *PF-KEY	149
Processing Based on Function-Key Names	150
System Variable *PF-NAME	150
Processing Data Outside an Active Window	151
System Variable *COM	151
Example Usage of *COM	151
Positioning the Cursor to *COM - %T* Terminal Command	152
Copying Data from a Screen	154
Terminal Commands %CS and %CC	154
Selecting a Line from Report Output for further Processing	154
Statements REINPUT/REINPUT FULL	156









Object-Oriented Processing	159
Natural Command Processor	159
Editors - General Information	160
Editors - General Information	160
Object Names	161
Split-Screen Mode	162
Split-Screen Commands	162
Editor Profile	164
General Information	164
Additional Options	166
Editor Defaults	167
General Defaults	168
Color Definitions	169
Direct Commands	170
User Exit USR0070P	171
Exit Profile Maintenance	171
Program Editor	172
Program Editor	172
Invoking the Program Editor	172
Top Information Line	173
Bottom Information Line	173
Editor Command Line	173
Editing a Program	173
Multiple Functions	173
Dynamic Conversion from Lower to Upper Case	174
Editor Commands	175
Editor Commands for Positioning	178
Line Commands	179
Special PF-Key Functions	181
Cursor-Sensitive Commands	182
The SCAN Commands	182
The SPLIT Command	182
The EDIT and LIST System Commands	183
The Exit Function	183
Data Area Editor	184
Data Area Editor	184
Invoking the Data Area Editor	184
Top Information Line	185
Bottom Information Line	185
Editor Command Line	185
Editing a Data Area	186
Editor Commands	188
Line Commands	190
Edit Fields	193
Special Commands Available within the Edit Fields Function	195
The ".E" Line Command with Control Variables	195
The Exit Function	196
Defining Globally Unique IDs in the Local and Global Data Area Editors	196
Map Editor	197
Map Editor	197
Components of the Map Editor	198
Summary of Map Creation	199
Step 1	199
Step 2	199
Step 3	199
Step 4	199

Invoking the Map Editor	200
Overview of Functions	202
Initializing a Map	206
Delimiters	207
Format	208
Context	210
Filler Characters	211
Editing a Map	212
Commands and Function Keys for Positioning	213
Line Commands	214
Field Commands	216
Defining Map Fields	218
Defining Fields Directly on the Screen	218
Selecting Fields from a User View or Data Definition	219
Using System Variables in a Map Definition	220
Extended Field Editing	221
HE Parameter Syntax:	224
Post Assignment Function	225
Array and Table Definition	226
Array Definition	226
Table Definition	230
Processing Rules	234
Field-Related Processing Rules	234
Function-Key-Related Processing Rules	235
Processing Rule Editing	236

User's Guide for Mainframes - Overview

This documentation introduces you to the use of Natural on mainframe computers. It also contains reference information on Natural editors.

Note that the Natural system commands are described in the Natural Command Reference documentation. System Commands are used to perform various system-related activities for example, saving a program, invoking an editor, logging on to another library.

- | | |
|---|---|
|  Fundamentals | Describes how to invoke and how to terminate a Natural session, online help, the main menus, and various other aspects of how to find your way in the Natural system; it also provides an overview of the Natural editors, system commands and terminal commands. |
|  Tutorial - Getting Started with Natural | Contains a series of tutorial sessions which introduce you to some of the basics of Natural programming. |
|  Tutorial - Using the Map Editor | Contains a series of tutorial sessions on how to use the Natural map editor for the creation of maps, processing rules and help routines. |
|  Designing User Interfaces | Provides information on components of Natural which you can use to design the user interfaces of your applications. |
|  Editors - General Information | Contains information that applies to all three Natural editors: the program editor, the data area editor and the map editor. |
|  Program Editor | Describes the program editor, which is used to create and maintain Natural programs, subprograms, subroutines, help routines, copycodes and texts. |
|  Data Area Editor | Describes the data area editor, which is used to create and maintain Natural local data areas, global data areas and parameter data areas. |
|  Map Editor | Describes the data area editor, which is used to create and maintain Natural maps and help maps. |

Fundamentals

This section covers the following topics:

- Components of Natural
 - Invoking Natural
 - Terminating Natural
 - Using the ENTER Key
 - Online Help System
 - Navigating within Natural
 - Main Natural Menus
 - Natural Editors
 - Terminal Commands
 - Asterisk Notation
-

Components of Natural

Natural is a complete environment for application development, offering all the functions you need to create an application:

- the Natural programming language;
- editors to create and maintain programs, maps, data areas and the other types of programming objects that make up a Natural application;
- a utility to create and maintain error messages to be issued by an application;
- various utilities for online testing and debugging of individual programs as well as entire applications;
- several other utilities for various purposes which you will find helpful when developing an application with Natural.

All components of an application can be created and compiled online. No batch or asynchronous compilation or link steps are required to create a Natural application.

With Natural applications you can access data that may be stored in Adabas databases as well as in a wide variety of other database systems.

Invoking Natural

Natural may be invoked for online or batch mode execution.

The way you invoke Natural depends on the specific TP monitor environment at your site. Ask your Natural administrator how to invoke Natural.

If Natural Security is installed, the access to some libraries as well as the use of some functions may be restricted. Ask your Natural administrator for details.

Terminating Natural

- Terminating an Online Session
- Terminating a Batch Session

Terminating an Online Session

A Natural online session can be terminated by any of the following:

- pressing PF3 or entering a period (.) in the command line of the Main Menu,
- entering the system command FIN,
- pressing CLEAR or an equivalent key,
- executing a Natural program which contains a TERMINATE statement.

The method of session termination may also be modified by the Natural administrator.

Terminating a Batch Session

A Natural batch mode session will be terminated when one of the following is encountered during the session:

- a FIN command in the input dataset,
- an end-of-input condition in the input dataset,
- a TERMINATE statement in a Natural program which is being executed.

Using the ENTER Key

To perform a particular Natural action, you enter the appropriate function code, command, etc., and then press ENTER.

So, if this documentation tells you to "enter a function code", this means, "type in the function code and press ENTER".

If a function requires that you press another key, this will be explicitly mentioned in this documentation.

Online Help System

Natural offers several types of online help:

- Each menu has a help option which can be invoked by entering a question mark (?) or by pressing the appropriate PF key as indicated in the PF-key lines on the screen (usually PF1).
- Each editor provides help information, which can be invoked from within the editor.
- When an error message is displayed, there is help available which provides a detailed explanation of the message: Help on Natural System Messages

When you enter the system command HELP or a question mark (?), or press the PF key to which the function "Help" is assigned, this will invoke the help system from which you can select the help you want.

To get information on a specific statement or system command, you enter in the command line HELP followed by the name of the statement/command (for example, HELP STOW).

Some Natural screens provide field-specific help, which you get by entering a question mark (?) in a field.

With the LASTMSG command, you can display additional information about the error situation which has occurred last. See LASTMSG in the Natural Command Reference documentation.

Help on Natural System Messages

The system messages issued by Natural begin with NAT followed by a four-digit number *nnnn*.

For each Natural system message, there is a *short text* and a *long text*:

- The *short text* is the one-line message which is displayed when the error occurs.
- The *long text* is an extended explanation of the error and the action to be taken.

To display the long text, you place the cursor in the message line (that is, the line in which the short text of the message is being displayed) and press the help key (by default PF1). Or you enter the system command "HELP *nnnn*" or "? *nnnn*" (*nnnn* being the error number).

For further information on help for error messages, see the system command HELP in the Natural Command Reference documentation.

Navigating within Natural

- Invoking a Function from a Menu
- Invoking a Function with a Command
- Leaving a Function
- PF Keys

You can invoke a Natural function either by selecting it from a menu or by entering a command.

Invoking a Function from a Menu

Every Natural menu screen offers you a list of functions.

- On some menu screens, there is an input field before each function listed. To invoke a function, mark the corresponding input field - either with the cursor or with any character.
- On some menu screens, a one-letter code is displayed before each function listed. To invoke a function, enter the corresponding code in the Code field provided on the screen.

Invoking a Function with a Command

All Natural screens provide a "command line". The command line is usually above the PF-key lines and looks as follows:

```
Command ===>
```

In this line, you can enter a Natural system command. With a system command, you can invoke a function directly, instead of having to "work your way" towards it along a number of menu screens.

You can also enter a system command in response to a NEXT or MORE prompt in the same way as in the command line of a menu screen.

If no menu screen is displayed, you will get a NEXT prompt, indicating that Natural is awaiting your next input.

During the execution of a program, Natural will display a MORE prompt to inform you that additional output is available. To display the additional output, you press ENTER. If you enter a system command in response to the MORE prompt, the program that is being executed will be stopped and the system command will be executed.

The input of Natural commands (system, line, editor, etc.) is **not** case-sensitive.

For further information, see the Natural Command Reference documentation.

Leaving a Function

To leave a Natural function, you enter a period (.) or press PF3 or PF12 (see below).

PF Keys

By default, the following functions are assigned to the following keys throughout Natural:

Key	Function	Explanation
PF1	Help	Invokes the online help system.
PF2	Menu	Invokes the Main Menu.
PF3	Exit	Leaves a function and applies the changes made previously.
PF12	Canc	Leaves a function and cancels the changes made previously.

Main Natural Menus

- Main Menu
- Development Functions
- Changing the Library
- Programming Modes
- Development Environment Settings
- Maintenance and Transfer Utilities
- Debugging and Monitoring Utilities
- Example Libraries
- Other Products

Main Menu

When you invoke Natural, the Main Menu will be displayed.

If you get a NEXT or MORE prompt instead of the Main Menu, enter the system command MAINMENU. The Natural Main Menu will then be displayed.

```
16:50:53          ***** NATURAL *****          2001-01-30
User SAG          - Main Menu -          Library SYSTEM

                Function

                _ Development Functions
                _ Development Environment Settings
                _ Maintenance and Transfer Utilities
                _ Debugging and Monitoring Utilities
                _ Example Libraries
                _ Other Products
                _ Help
                _ Exit NATURAL Session

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
                Help                Exit                                Canc
```

From the Main Menu, you can select one of the following functions:

Function	Explanation
Development Functions	Invokes a menu from which you can select various functions used to create and maintain programs, maps, data areas and the other components that make up a Natural application.
Development Environment Settings	Invokes a menu from which you can select various functions which allow you to display and modify various settings that affect your Natural session.
Maintenance and Transfer Utilities	Invokes a menu from which you can select various functions used to create and maintain certain objects or transfer them to another environment.
Debugging and Monitoring Utilities	Invokes a menu from which you can select various functions used to monitor your Natural applications and locate errors in their processing flow.
Example Libraries	Invokes a menu from which you can select various libraries containing example programs and user exits.
Other Products	Invokes a menu from which you can invoke several other Software AG products.

The position and color of the message line and PF-key lines on the main menu and its subordinate menus can be changed with the user exit USR2003 (which is provided in the library SYSEXT).

Development Functions

When you select "Development Functions" on the Natural Main Menu, the Development Functions menu is displayed:

16:51:14	***** NATURAL *****	2001-01-30
User SAG	- Development Functions -	Library SYSTEM
		Mode Structured
		Work area empty
	Code Function	
	C Create Object	
	E Edit Object	
	R Rename Object	
	D Delete Object	
	X Execute Program	
	L List Object(s)	
	S List Subroutines Used	
	? Help	
	. Exit	
	Code .. _ Type .. _	
		Name .. _____
Command ==>		
Enter-PF1---	PF2---PF3---	PF4---PF5---
PF6---	PF7---	PF8---
PF9---	PF10---	PF11---
PF12---		
Help	Exit	Canc

The functions listed on this menu are some of the functions you will need most frequently when you develop an application with Natural.

You can select a function from the Development Functions menu in three ways:

- **Input Fields**

You can enter the corresponding function code in the Code field.

To perform a function on a programming object which already exists, you enter the desired function code in the Code field and the name of the programming object in the Name field.

Some functions require that, in addition to entering the corresponding function code in the Code field, you enter an object type in the Type field. If you fail to do so, a window will automatically be displayed from which you can select an object type.

Once you are familiar with the object type codes displayed in the window, you can enter them directly in the Type field on the menu.

The various object types are described in the section Object Types of the Natural Programming Guide.

If you know the name of the object you wish to deal with, you can enter it in the Name field (without having to enter any Type).

If you invoke the function "Edit Object" or "List Object(s)" without specifying a name or object type, the current contents of the source work area will be displayed.

- **PF Keys**

You can press a PF key to invoke a function.

The PF-key lines at the bottom of the screen indicate which function is assigned to which key.

- **Command Line**

You can enter a Natural system command in the command line as described earlier under in the section Invoking a Function with a Command.

In general, the format of the commands corresponds to the Code/Name sequence. For example, to edit an existing program named PROGX, you would enter "E" in the Code field and PROGX in the Name field.

The equivalent system command to be entered in the Command line would be EDIT PROGX.

For further information, see the Natural Command Reference documentation.

The Development Functions menu provides the following functions:

Function	Explanation
Create Object	<p>With this function, you can create a new Natural programming object (program, map, data area, etc).</p> <p>You have to specify the type and the name of the object to be created. The appropriate editor will then be invoked: program editor, map editor, or data area editor.</p>
Edit Object	<p>With this function, you can modify the source code of an existing programming object.</p> <p>You have to specify the name of the object to be edited. The appropriate editor will then be invoked: program editor, map editor, or data area editor.</p> <p>If you do not remember the name, you can use the function "List Object(s)" (see below).</p>
Rename Object	<p>With this function, you can change the name of a programming object. This function is equivalent to the system command RENAME as described in the Natural Command Reference documentation.</p>
Delete Object	<p>With this function, you can delete one or more programming objects. This function is equivalent to the system command DELETE as described in the Natural Command Reference documentation.</p>
Execute Program	<p>With this function, you can execute a Natural object of type program. You have to specify the name of the program to be executed.</p> <p>Other object types cannot be executed by themselves, but must be invoked from another object.</p> <p>This function is equivalent to the system command EXECUTE as described in the Natural Command Reference documentation.</p>
List Object(s)	<p>This function allows you to select from a list the programming object you wish to edit.</p> <p>This function is equivalent to the system command LIST as described in the Natural Command Reference documentation.</p>
List Subroutines Used	<p>With this function, you can ascertain which programming objects in the current library use which external subroutines.</p> <p>This function is equivalent to the system command ROUTINES as described in the Natural Command Reference documentation.</p>

Changing the Library

In the top right-hand corner of the Development Functions menu is a Library field, which indicates the ID of your current library, that is, the current library where programming objects are stored and from which they are retrieved.

The library ID is in effect until you change it, or until the end of your Natural session. The default library ID assigned by Natural is "SYSTEM".

On the Development Functions menu, you can change libraries by overwriting the library ID displayed in the top right-hand corner with another library ID.

Generally, you can change libraries anywhere in Natural by entering the following system command in the command line:

LOGON *library-ID*

where *library-ID* is the ID of the library you want to access.

Programming Modes

Natural offers two ways of programming: reporting mode and structured mode. Generally, it is recommended to use structured mode exclusively, because it provides for more clearly structured applications. Therefore all explanations and examples in the Natural User's Guide for Mainframes refer to structured mode. Any peculiarities of reporting mode will not be taken into consideration. (For differences between the two modes, refer to the section Reporting Mode and Structured Mode in the Natural Programming Guide.)

In the top right-hand corner of the Development Functions menu is a Mode field, which indicates the programming mode currently in effect: "Structured" or "Reporting".

To change the mode, you overwrite the first position of the Mode field with an "S" (for structured mode) or an "R" (for reporting mode).

Development Environment Settings

When you select "Development Environment Settings" on the Natural Main Menu, a menu with the following functions is displayed:

Function	Description
Function-Key Settings	With this function, you can assign functions to PF keys to be used in your Natural session. Corresponding command: KEY
Compilation Settings	With this function, you can set various options that affect the way in which Natural programming objects are compiled. Corresponding command: COMPOPT
Session Parameter Settings	With this function, you can change the settings of various Natural session parameters. Session parameters are described in the Natural Parameter Reference documentation. Corresponding command: GLOBALS.
Profile Parameter Settings	With this function, you can change the settings of various Natural profile parameters. Profile parameters are described in the Natural Parameter Reference documentation and in Profile Parameter Usage in the Natural Operations for Mainframes documentation. The command SYSPARM invokes a utility of the same name. The SYSPARM utility is described in the Natural Utilities for Mainframes documentation. Corresponding command: SYSPARM.
Technical Session Information	This function displays various items of technical information on your Natural session. Corresponding command: TECH.
System File Information	This function displays the current definitions of the Natural system files. Corresponding command: SYSPROF.
Product Installation Information	This function displays a list of the products installed at your site and some information on these products. Corresponding command: SYSPROD.
Security Profile Information	This function displays the security profile currently in effect for you. (This function is only available if Natural Security is installed.) Corresponding command: PROFILE.

To invoke a function from the menu, you mark the corresponding input field - either with the cursor or with any character.

You can also invoke each function via a corresponding system command (as indicated in the table above). The system commands are described in the Natural Command Reference documentation.

For details on each function, refer to the description of the corresponding system command or utility.

Maintenance and Transfer Utilities

When you select "Maintenance and Transfer Utilities" on the Natural Main Menu, a menu with the following utilities is displayed:

Function	Description
Maintain Error Messages	With this utility, you create and maintain the messages you wish to issue in your Natural applications. Corresponding command: SYSERR (*).
Maintain DDMs	With this utility, you create and maintain the data definition modules (DDMs), that is, the logical definitions of the database files you wish to access in your Natural applications. For a detailed explanation of DDMs, see the section Database Access in the Natural Programming Guide. Corresponding command: SYSDDM (*).
Maintain Command Processors	With this utility, you create and maintain the command processors you wish to use in your Natural applications. Corresponding command: SYSNCP (*).
Maintain Remote Procedure Calls	With this utility, you create and maintain remote procedure calls, that is, provide the settings necessary to execute a Natural subprogram located on a remote server. Corresponding command: SYSRPC (*).
Transfer Objects to Other Libraries	With this utility, you can transfer Natural programming objects, error messages, DDMs and several other objects from one library to another. Corresponding command: SYSMAIN (*).
Transfer Objects to Other System Files	With this utility, you can transfer Natural programming objects, DDMs and error messages from one system file to another. Corresponding command: SYSUNLD (*).
Transfer Objects to Other Platforms	With this utility, you can transfer Natural programming objects, DDMs, error messages and Adabas FDTs from one hardware platform to another. Corresponding command: SYSTRANS (*).

* Each of these commands invokes a utility or application of a corresponding name. For a description of the utilities see the Natural Utilities for Mainframes documentation or refer to the list of utilities on the overview page of Natural for Mainframes.

To invoke a function from the menu, you mark the corresponding input field - either with the cursor or with any character.

You can also invoke each function via a corresponding system command (as indicated in the table above).

For details on each function, refer to the description of the corresponding system command or utility.

Debugging and Monitoring Utilities

When you select "Debugging and Monitoring Utilities" on the Natural Main Menu, a menu with the following utilities is displayed:

Function	Description
Debugging	With this utility, you can search for errors in the processing flow of programs. Corresponding command: TEST.
Logging of Database Calls	With this utility, you can log database commands. Corresponding command: TEST DBLOG.
Issuing Adabas Calls	With this utility, you can pass Adabas commands directly to the database. Corresponding command: SYSADA (*)
Buffer Pool Maintenance	With this utility, you can monitor the Natural buffer pool and adjust it to meet your requirements. Corresponding command: SYSBPM (*).
Editor Buffer Pool Maintenance	With this utility, you can monitor the buffer pool of the Software AG Editor and adjust it to meet your requirements. Corresponding command: SYSEDIT (*).
TP-Specific Monitoring	With this utility, you can monitor and control various TP-monitor-specific characteristics of Natural. Corresponding command: SYSTP (*).
Data Collection and Tracing	With this utility, you can collect monitoring and accounting data about the processing flow of a Natural application. Corresponding command: SYSRDC (*).
Error Information on Abnormal Termination	This function provides information for Software AG Technical Support required for error diagnosis. Corresponding command: DUMP.

You can also invoke each function via a corresponding system command (as indicated in the table above). The system commands are described in the Natural Command Reference documentation.

* Each of these commands invokes a utility or application of a corresponding name. For a description of the utilities see the Natural Utilities for Mainframes documentation or refer to the list of utilities on the overview page of Natural for Mainframes. SYSEDIT is described in the Natural Operations for Mainframes documentation.

To invoke a function from the menu, you mark the corresponding input field - either with the cursor or with any character.

You can also invoke each function via a corresponding system command (as indicated in the table above).

For details on each function, refer to the description of the corresponding system command or utility.

Example Libraries

When you select "Example Libraries" from the Natural Main Menu, a list of libraries containing example programs and user exits provided by Software AG will be displayed:

Library	Contents
SYSEXP	This library contains the example programs shown and referred to in the Natural Programming Guide.
SYSEXR	This library contains the example programs shown and referred to in the Natural Command Reference documentation and the Natural Statements documentation.
SYSEXP23	This library contains example programs which illustrate some of the new features provided with Version 2.3 of Natural.
SYSEXT	This library contains various Natural user exits; see also the system command SYSEXT as described in the Natural Command Reference documentation.
SYSEXTP	This library contains various Natural user exits for specific functions that apply only under certain TP monitors.

To display the contents of a library, you mark the corresponding input field - either with the cursor or with a character.

Other Products

When you select "Other Products" from the Natural Main Menu, a list will be displayed showing the Software AG products installed at your site which can be invoked via Natural and to which you have access.

To invoke a product, you mark the corresponding input field - either with the cursor or with a character.

Natural Editors

Natural provides three editors: the program editor, the data area editor, and the map editor.

The type of programming object to be edited determines the editor you will use. When you specify a programming object by name, the appropriate editor is automatically invoked.

- **Program Editor** - This editor is used to create and maintain programs, subroutines, subprograms, help routines, copy codes and texts.
- **Data Area Editor** - This editor is used to create and maintain global data areas, local data areas, and parameter data areas.
This editor has a columnar format that is designed for defining the data used in Natural programs or routines.
- **Map Editor** - This editor is used to create and maintain maps (screen layouts) referenced in a program's INPUT or WRITE statement.
The map editor allows direct manipulation of the fields used in an input or output map; the extended field editing feature facilitates the definition of fields; moreover, processing rules can be attached to fields in the map.

All editors can be operated in split-screen mode so that a portion of the screen can be used to display related objects. While creating or editing a map in the map editor, for example, you may have DDM fields displayed and transfer these fields into the map.

Examples for the use of editors are provided in the sections Tutorial - Getting Started with Natural and Tutorial - Using the Map Editor.

For further general information, see the section Editors - General Information.

Terminal Commands

Natural terminal commands are used to perform a wide variety of functions.

All terminal command begin with a percent sign (%).

The Natural Programming Reference documentation contains a description of each terminal command. See Terminal Command List or Terminal Commands Grouped by Function.

Note:

Within a program, you can assign terminal commands to function keys by using the SET KEY statement. With the system command KEY you can also assign terminal commands to functions keys. See also (Terminal Command) Key Assignments in the Natural Programming Reference documentation.

Asterisk Notation

Many Natural functions display lists of objects. Usually, these lists contain all objects available (for example, all objects of a given type, all objects in a given library). If you do not wish all objects to be listed, but only a certain range of objects, you may specify that range by using *asterisk notation*:

By specifying a parameter value followed by an asterisk (*) you will get a list of only those objects whose names (or IDs or whatever the parameter is) begin with that value. This option to enter a value followed by an asterisk is referred to as *asterisk notation*.

Example 1:

If you enter the system command DELETE without any parameters:

DELETE

you will get a list of all objects in the current library (you can then mark those which are to be deleted).

Example 2:

If you enter the system command DELETE as follows:

DELETE DAWG*

you will get a list of only those objects in the current library whose names begin with DAWG.

Tutorial - Getting Started with Natural

The sample sessions provided in this section are based on a program, a map, and a subprogram. Ask your Natural administrator to make these available to you. If you do not have access to copies of these objects, you will still be able to step through these sessions by creating them yourself.

The output screens provided in the sessions are merely meant as examples and may not correspond with your results. Also, behavior and appearance of Natural (screen layout, system messages, etc.) may differ from your environment as they depend on the system parameters set by your Natural administration.

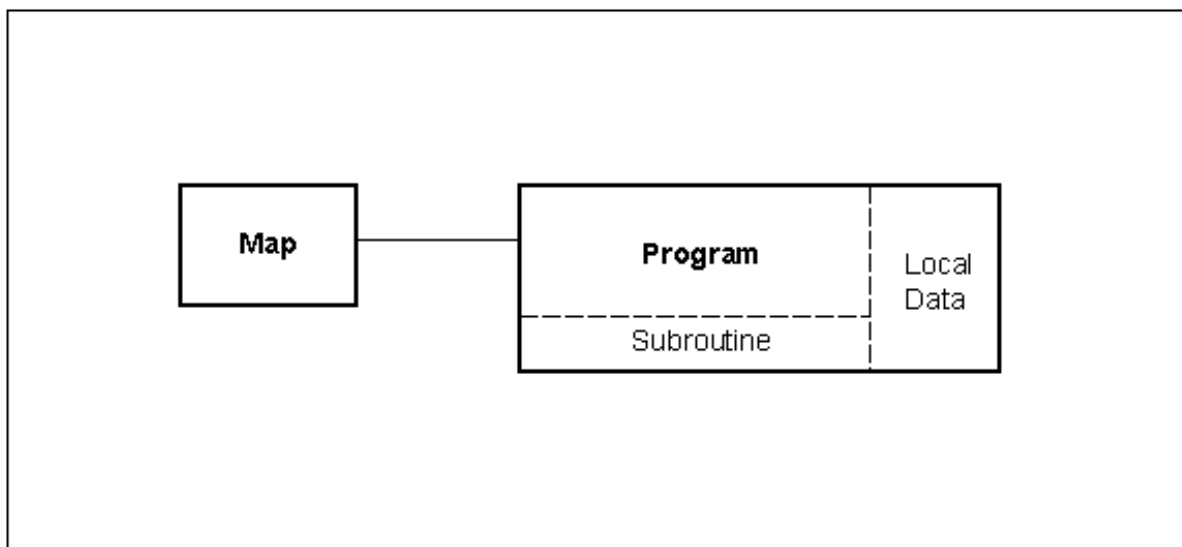
These sessions are **not** intended to provide an example of how an application should be built. Rather, each session is designed to provide a gradual exposure to specific features and build a context for introducing the next session's features. It is important that you work through the sessions in the sequence below. This tutorial approach does not parallel a typical application development cycle.

- Session 1 - Creating a Program and a Map
- Session 2 - Creating a Local Data Area
- Session 3 - Creating a Global Data Area
- Session 4 - Creating an External Subroutine
- Session 5 - Editing a Map
- Session 6 - Invoking a Subprogram

See also:

- Tutorial - Using the Map Editor

Session 1 - Creating a Program and a Map



In this session, you will use the *program editor* to create a Natural *program*. In this first session, the fields used in the program are defined as local data within the program. Moreover, an inline subroutine is contained within the program.

Also, you will use the *map editor* to create a *map* used by the program.

Step 1

Invoke Natural according to the procedures at your site. If you get a NEXT or MORE prompt, enter the command MAINMENU. The Natural Main Menu will be displayed:

```

16:50:53          ***** NATURAL *****          2001-01-30
User SAG          - Main Menu -          Library SYSTEM

                Function

        _ Development Functions
        _ Development Environment Settings
        _ Maintenance and Transfer Utilities
        _ Debugging and Monitoring Utilities
        _ Example Libraries
        _ Other Products
        _ Help
        _ Exit NATURAL Session

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help          Exit                                Canc

```

Step 2

Both Natural system programs and user-written applications are stored in *libraries*. It may be necessary to move from one library to another in order to perform a maintenance function or work on a different application.

The field "Library" in the top right-hand corner shows the ID of the current library. To move to another library, enter the command "LOGON *library-ID*" (*library-ID* being the ID of the library you want to access) in the Command line. If you have access to a library that contains copies of the sample programs used in these sessions, enter the *library-ID* of that library. By default, sample programs are provided in the system library SYSEXP; ask your Natural administrator for details.

Step 3

On the Natural Main Menu, select "Development Functions". The Development Functions menu will be displayed:

16:51:14	***** NATURAL *****	2001-01-30
User SAG	- Development Functions -	Library SYSTEM
		Mode Reporting
		Work area empty
Code	Function	
C	Create Object	
E	Edit Object	
R	Rename Object	
D	Delete Object	
X	Execute Program	
L	List Object(s)	
S	List Subroutines Used	
?	Help	
.	Exit	
Code .. _	Type .. _	
	Name .. _____	
Command ==>		
Enter-PF1---	PF2---PF3---	PF4---PF5---
PF6---	PF7---	PF8---
PF9---	PF10---	PF11---
PF12---		
Help	Exit	Canc

You must be operating in *structured mode* to work through these sessions. If your session is in *reporting mode*, change the mode by entering an "S" in the first position of the "Mode" field. (If you should not be able to change the mode in this manner - it is dependent on your site - contact your Natural administrator for assistance.)

Step 4

In the course of developing application systems, Natural objects can be created and modified from the Development Functions menu.

If you wish to get help information about the functions on this menu, enter a question mark (?) in the Code field.

Below the list of functions, there are three input fields: "Code", "Type", and "Name". Further below is the command line. You can perform a Natural function either by entering the appropriate values in the input fields, or by entering a system command in the command line.

For example, to invoke the program editor to edit an existing program PGM01, you would enter the following in the input fields:

?		Help	
.		Exit	
Code ..	E	Type ..	_
Name ..	PGM01_____		
Command ==>			
Enter-PF1---	PF2---	PF3---	PF4---
Help	Exit	PF5---	PF6---
		PF7---	PF8---
		PF9---	PF10---
		PF11---	PF12---
			Canc

The equivalent system command, entered in the command line, would be:

?		Help	
.		Exit	
Code ..	_	Type ..	_
Name ..	_____		
Command ==> EDIT PGM01			
Enter-PF1---	PF2---	PF3---	PF4---
Help	Exit	PF5---	PF6---
		PF7---	PF8---
		PF9---	PF10---
		PF11---	PF12---
			Canc

If an object already exists (such as, program PGM01 in the previous example), it is not necessary to specify its type in the "Type" field; this is because each object in a Natural library, regardless of its type (program, map, subroutine, etc.), must have a unique name. When you create a new object, you have to specify the type of the object in the "Type" field.

Once you have become familiar with the sequence of menu screens, you will be able to go directly to the screen you want by issuing a system command. You may enter a system command on every Natural screen which provides a command line.

Step 5

On the Development Functions menu, enter the code "E". The Natural *program editor* will be displayed:

```

>                                     > + Program                               Lib SYSTEM
All  .....1.....2.....3.....4.....5.....6.....7..
0010
0020
0030
0040
0050
0060
0070
0080
0090
0100
0110
0120
0130
0140
0150
0160
0170
0180
0190
0200
.....1.....2.....3.....4.....5..... S 0      L 1

```

- If you have access to a copy of program PGM01, enter the command READ PGM01 in the command line at the top of the editor screen. Make sure that the program matches program PGM01 shown on the following page.
- If you do not have access to a copy of PGM01, type in the program as illustrated in the following section. (If the source work area is not empty, enter the command CLEAR in the command line at the top of the editor screen.) As you fill up a screen, type the command ADD in the command line for more blank lines. When complete, enter the command SAVE PGM01 in the command line to store the program.

Note:

To return to the beginning of the program, enter the command TOP in the editor's command line, to go to the end of the program, enter the command BOT.

Program PGM01:

```

* Example Program 'PGM01' for User's Guide Tutorial
* -----
DEFINE DATA
  LOCAL
  01 #NAME-START      (A20)
  01 #NAME-END        (A20)
  01 #MARK            (A1)
  01 EMPLOYEES-VIEW VIEW OF EMPLOYEES
    02 PERSONNEL-ID (A8)
    02 NAME         (A20)
    02 DEPT         (A6)
    02 LEAVE-DUE    (N2)
END-DEFINE
*
REPEAT
*
INPUT USING MAP 'MAP01'
IF #NAME-START = ' .'

```

```
        ESCAPE BOTTOM
END-IF
MOVE #NAME-START TO #NAME-END
*
RD1. READ EMPLOYEES-VIEW BY NAME
        STARTING FROM #NAME-START
        THRU #NAME-END
IF LEAVE-DUE >= 20
        PERFORM MARK-SPECIAL-EMPLOYEES
    ELSE
        RESET #MARK
END-IF
    DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=20' #MARK
END-READ
*
IF *COUNTER (RD1.) = 0
    REINPUT 'PLEASE TRY ANOTHER NAME'
END-IF
*
END-REPEAT
*
DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
    MOVE '*' TO #MARK
END-SUBROUTINE
*
END
```

Once you reached this point (either by writing the program yourself or by reading it into the editor), note the following aspects of this example of a Natural program:

- The first statement must always be a DEFINE DATA statement. All variables to be used in the program must be defined in this initial DEFINE DATA statement. In this example program, the variables are defined in a DEFINE DATA LOCAL statement; that is, the data are defined *internal* to the program.
- All statements which initiate a logical construct or processing loop (DEFINE DATA, REPEAT, IF, READ) must be ended with a corresponding END-... statement (END-DEFINE, END-REPEAT, END-IF, END-READ).
- The READ statement is marked with a so-called *statement label*, namely "RD1.". Via this label it is possible to reference the statement at a later point in the program (see last IF statement).

When this program is executed, a screen appears, prompting the user to enter a name. The EMPLOYEES file is searched to locate all employees with that name. Then a report is displayed which includes the Name, Department and Leave Due of each employee with that name. Those employees who have more than 20 days leave due are marked with an asterisk (*).

The prompting screen is invoked via the INPUT USING MAP statement. The report is formatted according to information in the DISPLAY statement. The processing required to show which employees have more than 20 days leave is handled in the portion of the program starting with "IF LEAVE-DUE ...". Those with 20 or more days of leave due have an asterisk in the final report as a result of processing in the PERFORM statement and the DEFINE SUBROUTINE statement.

Step 6

The program contains an INPUT USING MAP statement which invokes a map named MAP01. This map is yet to be created.

In the program editor's command line, enter a period (.) to return to the Development Functions menu. On the Development Functions menu, enter the function code "E" (for Edit Object) in the Code field and "M" (for Map) in the Type field:

.		Exit
Code ..	E	Type .. M
	Name ..	_____
Command ==>		
Enter-PF1---	PF2---	PF3---
PF4---	PF5---	PF6---
PF7---	PF8---	PF9---
PF10---	PF11---	PF12---
Help	Exit	Canc

The Edit Map menu will be displayed:

17:55:11	***** NATURAL MAP EDITOR *****		2001-01-30
User SAG	- Edit Map -		Library SYSTEM
Code	Function		
----	-----		
D	Field and Variable Definitions		
E	Edit Map		
I	Initialize new Map		
H	Initialize a new Help Map		
M	Maintenance of Profiles & Devices		
S	Save Map		
T	Test Map		
W	Stow Map		
?	Help		
.	Exit		
Code ..	I	Name ..	Profile .. SYSPROF_

Command ==>			
Enter-PF1---	PF2---	PF3---	PF4---
PF5---	PF6---	PF7---	PF8---
PF9---	PF10---	PF11---	PF12---
Help	Exit	Test	Edit

Step 7

Enter the code "I" (Initiate a New Map) and the name MAP01. The Define Map Settings For Map screen will be displayed:

17:56:47				Define Map Settings for MAP		2001-01-30	
Delimiters				Format		Context	
-----				-----		-----	
Cls	Att	CD	Del	Page Size 23	Device Check _____
T	D		BLANK	Line Size 79	WRITE Statement	_____
T	I		?	Column Shift	... 0 (0/1)	INPUT Statement	X
A	D		_	Layout _____	Help	_____
A	I)	dynamic N (Y/N)	as field default	N (Y/N)
A	N		^	Zero Print N (Y/N)		
M	D		&	Case Default	... UC (UC/LC)		
M	I		:	Manual Skip N (Y/N)	Automatic Rule Rank	1
O	D		+	Decimal Char	Profile Name SYSPROF
O	I		(Standard Keys	.. N (Y/N)		
				Justification	.. L (L/R)	Filler Characters	
				Print Mode _	-----	
				Control Var _____	Optional, Partial
						Required, Partial
						Optional, Complete	...
						Required, Complete	...
Apply changes only to new fields?				N (Y/N)			
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---							
Help		Quit		Let			

Step 8

In the Filler Characters section of the screen, enter an underscore (_) after each of the options:

Justification .. L (L/R)	Filler Characters
Print Mode _	-----
	Optional, Partial _
Control Var _____	Required, Partial _
	Optional, Complete ... _
Apply changes only to new fields? N (Y/N)	Required, Complete ... _
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---	
Help	Quit
	Let

The map will use this character whenever a field has empty positions to fill within a field. (Delimiter characters can also be modified on this screen, however for these sessions, they are left unchanged.)

Press ENTER again. The map editor screen will be displayed in split-screen format:

```

Ob  _
.
.
.
.
.
.
001  --010---+-----+-----+---030---+-----+-----+---050---+-----+-----+---070---+-----

Ob D CLS ATT DEL      CLS ATT DEL
.   T  D   Blnk      T  I   ?
.   A  D   _         A  I   )
.   A  N   ^         M  D   &
.   M  I   :         O  D   +
.   O  I   (
.

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -    +    Full  <    >    Let

```

The top portion of the screen will not be used in this exercise. The bottom section is the editing area in which you will create the map.

During the creation of the map, you will use a number of map editor *line commands* and *field commands*.

- A line command begins with two periods (..). You enter it at the beginning of a line, and it applies to the whole line in which you enter it.
- A field command begins with one period (.). You enter it at the beginning of a field, and it applies only to the field in which you enter it.

(The sessions in the section Tutorial - Using the Map Editor also show you how to apply these commands to more than one line/field at a time.)

Step 9

Move the cursor to the second line of the editing area and type in the following:

Please enter starting name :X(20)

When you press ENTER, the screen will look as follows:

```

Ob _                               Ob D CLS ATT DEL      CLS ATT DEL
.                                .   T  D   Blnk      T  I   ?
.                                .   A  D   _         A  I   )
.                                .   A  N   ^         M  D   &
.                                .   M  I   :         O  D   +
.                                .   O  I   (
.                                .
001  --010---+-----+-----030---+-----+-----050---+-----+-----070---+-----

Please enter starting name :XXXXXXXXXXXXXXXXXXXXX

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help  Mset  Exit  Test  Edit  --    -    +    Full  <    >    Let

```

If the letters you typed in are automatically converted to upper case, press PF3 to return to the Edit Map menu and enter "%L" in the command line.

Step 10

Type in the line command ".C" (Center) over the first three positions of the line that contains "Please enter starting name" as shown below:

```

Ob _                               Ob D CLS ATT DEL      CLS ATT DEL
.                                .   T  D   Blnk      T  I   ?
.                                .   A  D   _         A  I   )
.                                .   A  N   ^         M  D   &
.                                .   M  I   :         O  D   +
.                                .   O  I   (
.                                .
001  --010---+-----+-----030---+-----+-----050---+-----+-----070---+-----

..case enter starting name :XXXXXXXXXXXXXXXXXXXXX

```


As a result, the line will be centered:

```

Ob _                               Ob D CLS ATT DEL      CLS ATT DEL
.                               .      T  D   Blnk      T  I   ?
.                               .      A  D   _          A  I   )
.                               .      A  N   ^          M  D   &
.                               .      M  I   :          O  D   +
.                               .      O  I   (
.                               .
001  --010---+-----+-----030---+-----+-----050---+-----+-----070---+-----

Please enter starting name :XXXXXXXXXXXXXXXXXXXXX

```

Step 11

Move the cursor to the end of the new line, leave a space after the field and type in the following text:

(. to exit)

The screen will now look as follows:

```

Ob _                               Ob D CLS ATT DEL      CLS ATT DEL
.                               .      T  D   Blnk      T  I   ?
.                               .      A  D   _          A  I   )
.                               .      A  N   ^          M  D   &
.                               .      M  I   :          O  D   +
.                               .      O  I   (
.                               .
001  --010---+-----+-----030---+-----+-----050---+-----+-----070---+-----

Please enter starting name :XXXXXXXXXXXXXXXXXXXXX (. to exit)

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help  Mset  Exit  Test  Edit  --    -    +    Full  <    >    Let

```

Step 12

Type in the field command ".E" (for extended field editing) over the first two positions of the field "XXXXXXXXXXXXXXXXXXXX", as shown below:

Ob _	Ob D	CLS	ATT	DEL	CLS	ATT	DEL
.	.	T	D	Blk	T	I	?
.	.	A	D	_	A	I)
.	.	A	N	^	M	D	&
.	.	M	I	:	O	D	+
.	.	O	I	(
.	.						
001	--010	----	-----	----	-----	----	-----

Please enter starting name .XXXXXXXXXXXXXXXXXXXX (. to exit)

This will invoke the extended field editing section for the field marked with the command:

Fld #001	Fmt A20

AD= MIT'_'_____	ZP= OFF
AL= _____	CD= ____
PM= ____ DF=	DY= _____
EM= _____	
001	--010

Please enter starting name .XXXXXXXXXXXXXXXXXXXX (. to exit)

The field "Fld" in the upper left corner contains the field name "#001". This number has been assigned automatically by Natural.

Overwrite "#001" with the field name "#NAME-START":

Fld #NAME-START	Fmt A20

AD= MIT'_'_____	ZP= OFF
AL= _____	CD= ____
PM= ____ DF=	DY= _____
EM= _____	
001	--010

Press PF3 to leave the extended field editing section.

Step 13

The map is now complete.

Press PF4 to test the map. The map will be displayed in the form it which it will be displayed on the screen it is invoked via the INPUT USING MAP statement in the program PGM01:

Please enter starting name _____ (. to exit)
--

Press PF3 to stop testing and return to the map editing screen.

Step 14

Press PF3 again to return to the Edit Map menu.

Enter the code "W". The map is now stowed (that is, stored in source form and in object form) and ready to be used by the program.

To return to the Development Functions menu, enter a period (.) in the Code field.

Step 15

In the command line of the Development Functions menu, enter the command EDIT PGM01.

Then enter the command RUN in the command line at the top of the program editor. This command compiles and executes the program PGM01.

A screen will be displayed requesting you to enter a name. For demonstration purposes, enter the name JONES.

Based on this name, the program will produce the following output report:

Page	1			00-11-30 12:45:56
NAME	DEPARTMENT CODE	LEAVE DUE	>=20	
-----	-----	-----	----	
JONES	SALE30	25	*	
JONES	MGMT10	34	*	
JONES	TECH10	11		
JONES	MGMT10	18		
JONES	TECH10	21	*	
JONES	SALE00	30	*	
JONES	SALE20	14		
JONES	COMP12	26	*	
JONES	TECH02	25	*	

Step 16

Keep pressing ENTER until you get to the map input screen again. When the program asks you again to enter a name, enter a period (.) and delete the remaining characters from the input field. You will be returned to the program editor.

Step 17

Whenever you issue a CHECK, RUN or STOW command, the program is checked for syntax errors that would keep the program from being processed.

To introduce such an error, move the cursor to the following statement line:

```
DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=20' #MARK
```

Delete the apostrophe after "20". The line now look as follows:

```
DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=20 #MARK
```

In the command line, enter the command CHECK. The error message "Text string must begin and end on the same line." will appear:

```

>                                     > + Program      PGM01      Lib SYSEXP
....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
0350          BY NAME
0360          STARTING FROM #NAME-START
0370          THRU #NAME-END
0380 *
0390  IF LEAVE-DUE >=20
0400      PERFORM MARK-SPECIAL-EMPLOYEES
0410  ELSE
0420      RESET #MARK
0430  END-IF
0440 *
E 0450  DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=20 #MARK
0460 *
0470  END-READ
0480 *
0490  IF *COUNTER (RD1.) = 0
0500      REINPUT 'PLEASE TRY ANOTHER NAME'
0510  END-IF
0520 *
0530  END-REPEAT
0540 *
....+....1....+....2....+....3....+....4....+....5....+.... S 59   L 35
NAT0305 Text string must begin and end on the same line.

```

Natural requires that a text string (in this case '>=20') must be begun and closed on the same statement line; the beginning and closing of a text string must be indicated by apostrophes. When the closing apostrophe is deleted, this condition is no longer met.

Note:

If you wish to get more information explaining the meaning of the error message, you can enter a question mark (?) and the error number of the message in the command line to invoke the help system, for example, "? NAT0305".

Step 18

Type in the missing apostrophe again. Then enter the command CHECK in the command line again to make sure the program is now correct.

Then enter the command RUN in the command line. When the program has run successfully, return to the program editor again by entering a period (.).

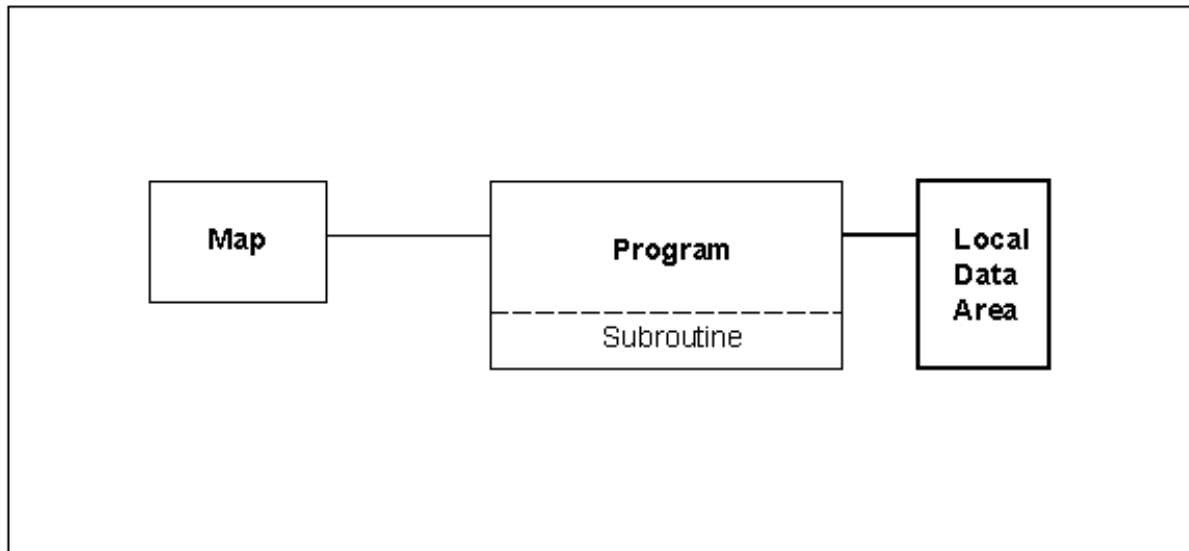
Step 19

Enter the command STOW in the command line to compile the program and save both source and object form.

Then enter a period (.) in the command line to return to the Development Functions menu.

End of Session 1.

Session 2 - Creating a Local Data Area



In Session 1, the fields used by the program were defined within the DEFINE DATA statement in the program itself. It is also possible to place the field definitions in a *local data area* outside the program, with the program's DEFINE DATA statement referencing that local data area by name.

In this session, the information in the DEFINE DATA statement will be relocated to a local data area outside the program. In subsequent sessions some of this information can be used as the basis of a global data area shared by a program and an external subroutine. As you will see in Sessions 3 and 4, an important advantage of data areas is to allow a program and its external subroutine to share the same data in a single data area.

Several program editor commands will be used in this and the following sessions. See the section Editor Commands for a detailed explanation.

Step 1

On the Development Functions menu, enter the code "E" and object type "L" to create a *local data area*.

The *data area editor* will be invoked with the object type set to "Local":

```

Local                               Library SYSTEM                               DBID  10 FNR  32
Command                               > +
I T L Name                           F Leng Index/Init/EM/Name/Comment
All - ----- - -----

```

In this editor, you can define the data areas to be used by Natural programs or routines. A basic outline of the fields on this screen is provided below:

Field	Explanation
I	This is an <i>information</i> field used by the editor to indicate the presence of a definition error ("E"), or to indicate other information on the variable (initial value, edit mask definition, etc.) You cannot modify information in this field.
T	This field indicates the <i>type</i> of the variable. For example, some fields in the data area to be created in this session will be part of a view, which will be indicated by a "V" in this column. Other object types include groups, data blocks, multiple-value fields, periodic groups, and constants.
L	In this field you specify the hierarchical <i>level</i> of the variable (1 to 9). A variable which is not within a hierarchical structure is assigned a level 1 designation.
Name	In this field you specify the <i>name</i> of the variable (or block or view).
F	In this field you specify the <i>format</i> of the variable.
Leng	In this field you specify the <i>length</i> of the variable.
Index/...	This field is used for array definition, initial value assignment, edit mask information, originating view name (for fields derived from a view), or a comment.

Step 2

In the command line of the data area editor, enter the command "SPLIT P PGM01". This puts the editor into split-screen mode with program PGM01 appearing in the lower half of the screen and the data area (as yet blank) in the upper half.

To scroll forward in the lower half of the screen, enter the command "SPLIT +" in the command line of the data area editor ("SPLIT -" scrolls backward). Enter the command again until the DEFINE DATA LOCAL statement and at least three statement lines below it are displayed on the screen:

Local	Library	SYSTEM	DBID	10	FNR	32
Command						> +
I T L	Name	F	Leng	Index/Init/EM/Name/Comment		
All	-	-----	-	----	-----	-----
-----						S 0 L 1
Program	PGM01	Library	SYSTEM			
0120	DEFINE DATA					
0130	LOCAL					
0140	01 #NAME-START		(A20)			
0150	01 #NAME-END		(A20)			
0160	01 #MARK		(A1)			
0170	01 EMPLOYEES-VIEW VIEW OF EMPLOYEES					

Step 3

Using the program as a reference, copy the definitions of the first three variables from the DEFINE DATA statement into the data area editor screen. In column "L" enter the level number "1" before each variable; in column "Name" enter the variable names; in column "F" enter their formats, and in column "Leng" their lengths.

The screen should now look as follows:

```

Local                               Library SYSTEM                               DBID  10 FNR  32
Command                               > +
I T L Name                           F Leng Index/Init/EM/Name/Comment
All - -----
    1 #NAME-START                     A   20
    1 #NAME-END                       A   20
    1 #MARK                           A    1

----- S 3      L 1
Program      PGM01      Library SYSTEM
    0120      DEFINE DATA
    0130      LOCAL
    0140      01 #NAME-START          (A20)
    0150      01 #NAME-END            (A20)
    0160      01 #MARK                (A1)
    0170      01 EMPLOYEES-VIEW VIEW OF EMPLOYEES

```

Each of the three variables has a "#" as the initial character. This character is used to identify user-defined variables; that is, to distinguish them from database fields.

Enter the command **CHECK** to see if everything you typed in as part of the new local data area is correct. This command ends split-screen mode and you return to the full-screen mode.

Step 4

The local data area is not yet complete. You can read the other variables you need directly from a view into the data area editor.

In the line below the variables you have already defined, enter the command ".V (EMPLOYEES)" (starting in column "T"). The view EMPLOYEES will be displayed:

Local	Library	SYSTEM	DBID	10	FNR	32
View	EMPLOYEES					
I T L	Name	F	Leng	Index/Init/EM/Name/Comment		
2	PERSONNEL-ID	A	8			
G 2	FULL-NAME					
3	FIRST-NAME	A	20			
3	MIDDLE-I	A	1			
3	NAME	A	20			
2	MIDDLE-NAME	A	20			
2	MAR-STAT	A	1			
2	SEX	A	1			
2	BIRTH	D				
2	NJBIRTH	I	2			
G 2	FULL-ADDRESS					
M 3	ADDRESS-LINE	A	20	(1:191) /* MU-FIELD		
3	CITY	A	20			
3	ZIP	A	10			
3	POST-CODE	A	10			
3	COUNTRY	A	3			
G 2	TELEPHONE					

SYSGDA 4461: Mark fields to incorporate into data area.						

Step 5

From this view (DDM), select the fields which you want to include in the local data area by marking them with any character in column "I". In this case, mark the fields PERSONNEL-ID, NAME, DEPT and LEAVE-DUE. DEPT and LEAVE-DUE are not in the first section of the view; to scroll forward within the view, keep pressing ENTER until the section which contains DEPT and LEAVE-DUE is displayed.

After you have marked all the fields indicated, continue pressing ENTER until the local data area - which now includes the fields you have selected from the EMPLOYEES view - is displayed again:

Local	Library	SYSTEM	DBID	10	FNR	32
Command						> +
I	T	L	Name	F	Leng	Index/Init/EM/Name/Comment
All	-			-		
	1		#NAME-START	A	20	
	1		#NAME-END	A	20	
	1		#MARK	A	1	
V	1		EMPLOYEES-VIEW			EMPLOYEES
	2		PERSONNEL-ID	A	8	
	2		NAME	A	20	
	2		DEPT	A	6	
	2		LEAVE-DUE	N	2.0	
----- S 8 L 1						
SYSGDA 4462: 4 field(s) of view EMPLOYEES included.						

Step 6

The local data area is now complete. To check if it contains any errors, enter the command CHECK in the command line of the editor.

If it contains any errors, correct them and repeat the CHECK.

Then enter the command STOW LDA01 in the command line. The local data area is now compiled and stored in source and object form under the name LDA01.

Note that a data area must be compiled and stored in object form before any program referencing that data area can be compiled.

Step 7

Now program PGM01 must be edited to reference the local data area LDA01.

In the command line of the data area editor, enter the command EDIT PGM01. This invokes the program editor and reads the program PGM01 into the work area of the program editor.

Step 8

Now that the variables are defined in the local data area LDA01, the DEFINE DATA statement has to be changed from actually containing the definition of variables to merely referencing the data area in which the variables are defined.

Enter the command ".D" at the beginning of each line that contains a variable definition within the DEFINE DATA statement, as shown below:

```

>
> + Program      PGM01      Lib SYSTEM
....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
0010 * Example Program: PGM01
0020 * Function: Demonstrate Natural
0030 * -----
0040 DEFINE DATA
0050   LOCAL
0060   .d 01 #NAME-START          (A20)
0070   .d 01 #NAME-END          (A20)
0080   .d 01 #MARK              (A1)
0090   .d 01 EMPLOYEES-VIEW VIEW OF EMPLOYEES
0100   .d   02 PERSONNEL-ID      (A8)
0110   .d   02 NAME              (A20)
0120   .d   02 DEPT              (A6)
0130   .d   02 LEAVE-DUE         (N2)
0140 END-DEFINE

```

When you press ENTER, these lines are deleted from the program.

Then enter the following after the word LOCAL:

```
USING LDA01
```

The program should now look as follows:

Program PGM01:

```
* Example Program 'PGM01' for User's Guide Tutorial
* -----
DEFINE DATA
  LOCAL USING LDA01
END-DEFINE          *
REPEAT
*
INPUT USING MAP 'MAP01'
IF #NAME-START = '.'
  ESCAPE BOTTOM
END-IF
MOVE #NAME-START TO #NAME-END
*
RD1. READ EMPLOYEES-VIEW
  BY NAME
    STARTING FROM #NAME-START
    THRU #NAME-END
  IF LEAVE-DUE >= 20
    PERFORM MARK-SPECIAL-EMPLOYEES
  ELSE
    RESET #MARK
  END-IF
  DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=20' #MARK
END-READ
*
  IF *COUNTER (RD1.) = 0
    REINPUT 'PLEASE TRY ANOTHER NAME'
  END-IF
*
END-REPEAT
*
DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
  MOVE '*' TO #MARK
END-SUBROUTINE
*
END
```

Step 9

You can enter comments into a program to document the changes you made to the program. Comments help anyone editing or maintaining a source program, and are ignored in processing.

You mark a line as a comment line by entering either an asterisk and a blank (*) or two asterisks (**) at the beginning of the line. In the rest of the line you can then enter any comment. If you wish to append a comment to a line containing a statement, you enter the character string blank-slash-asterisk (/*); anything to the right of this character string will then be considered a comment and ignored at execution.

In the second line of the program, enter the command ".I(1)" to insert an empty line. In the empty line, add some comment to indicate that this program has been updated, as shown in the example below.

Example:

```
* Example Program 'PGM01' for User's Guide Tutorial* PROGRAM NOW USES A LOCAL DATA AREA
* -----
DEFINE DATA
  LOCAL USING LDA01 /* This comment is for demonstration purposes only.
END-DEFINE
...
```

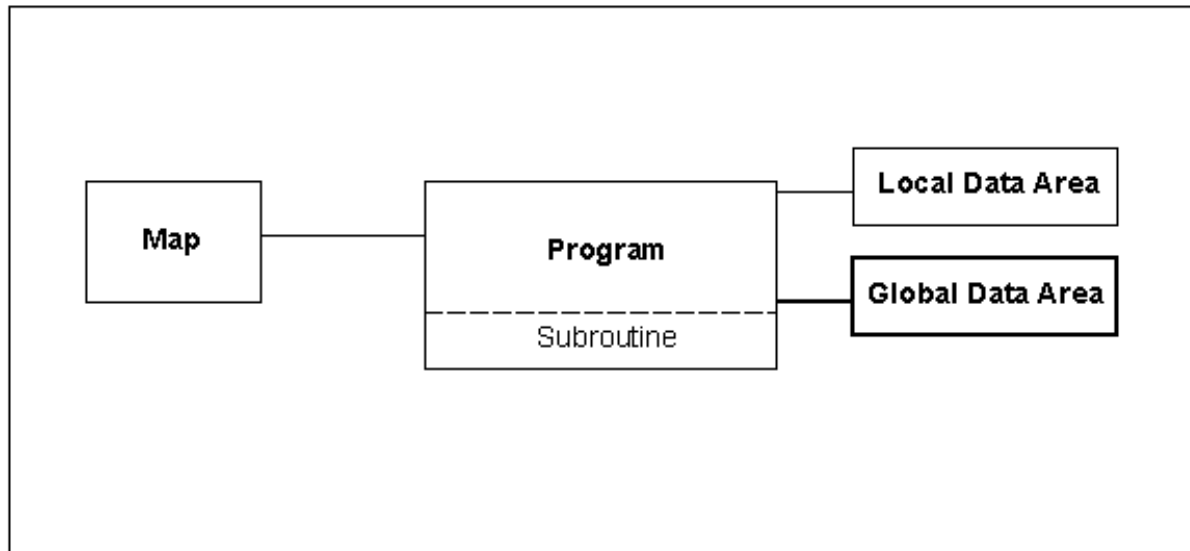
The subsequent sessions in this section show further examples of comments, which you can add to keep track of changes you have made.

Step 10

CHECK the program and correct any errors. RUN the program to confirm that the results are the same as when the DEFINE DATA statement did not reference a local data area. STOW the program so that it will be available for Session 3.

End of Session 2.

Session 3 - Creating a Global Data Area



In Natural, data can be defined in a single location outside any particular program or routine. Data defined in such a *global data area* can then be shared by multiple programs/routines.

In this session, you will create a global data area. In addition, the local data area created in Session 2 will be modified. Moreover, the program has to be modified to reference not only the local data area, but also the new global data area.

Step 1

On the Development Functions menu, enter the code "E" and name LDA01. This invokes the data area editor and reads the local data area LDA01 into the work area of the editor.

Step 2

To create the new data area, save the contents of the work area under a different name: in the command line of the editor, enter the command SAVE GDA01.

Then enter the command READ GDA01 to read the newly created data area into the work area. As it is identical to the original one, only the name displayed at the top of the data area editor will change from LDA01 to GDA01.

Step 3

Then enter the command SET TYPE GLOBAL. As a result, you will now be editing the data area in the editor as a *global data area*.

Enter the line command ".D" to delete the first two lines (#NAME-START and #NAME-END). The data area now looks as follows:

Global	GDA01	Library	SYSTEM	DBID	10	FNR	32
Command							> +
I T L	Name		F	Leng	Index/Init/EM/Name/Comment		
All	-	-----	-	-----	-----		
	1	#MARK	A	1			
V	1	EMPLOYEES-VIEW			EMPLOYEES		
	2	PERSONNEL-ID	A	8			
	2	NAME	A	20			
	2	DEPT	A	6			
	2	LEAVE-DUE	N	2.0			
-----							S 6 L 1

Keep in mind that a data area must be compiled and stored in object form before any program referencing that data area can be compiled and executed.

In the command line, enter the command STOW. GDA01 is now compiled and stored in source and object form.

Step 4

Now some variables must be removed from the local data area, because they are now defined in the new global data area.

Enter the command READ LDA01 in the command line to read the local data area LDA01 into the editor.

With the line command ".D", delete from LDA01 all variables which are now also defined in GDA01; that is, everything except the two variables #NAME-START and #NAME-END. The modified data area now looks as follows:

Local LDA01		Library SYSTEM	DBID 10 FNR 32	
Command			>+	
I T L	Name	F	Leng	Index/Init/EM/Name/Comment
All	-	-	-	-
1	#NAME-START	A	20	
1	#NAME-END	A	20	
-----				S 2 L 1

In the command line, enter the command STOW.

The modified local data area is now ready to be referenced in the program.

Step 5

In the command line of the data area editor, enter the command EDIT PGM01. As PGM01 is stored as an object of type program, the program editor will automatically be invoked.

As the defined data are now located in two data areas, the DEFINE DATA statement in the program must now reference the global data area GDA01 as well as the local data area LDA01.

In the line which contains DEFINE DATA, enter the command ".I(1)" to insert one empty line. Type in GLOBAL USING GDA01 in the empty line.

The DEFINE DATA statement block should now look as follows:

```
...  
DEFINE DATA  
    GLOBAL USING GDA01  
    LOCAL  USING LDA01  
END-DEFINE  
...
```

Step 6

Moreover, we will change the instructions for the output produced by the program: we will add a WRITE TITLE statement and modify the DISPLAY statement.

Using the program example below, modify the program to include the WRITE TITLE statement (above the DISPLAY statement) and the new format in the DISPLAY statement. Use the line command ".I" to create the empty lines you need for these modifications. Also, add some comments at the top of the program to indicate the changes you have made.

The WRITE TITLE statement used in this program produces a multiple line title in the resulting report. The slash (/) notation causes a line advance. As nothing else is specified, the title lines will be displayed centered and not underlined.

The revised program - particularly the DEFINE DATA, WRITE TITLE and DISPLAY statements blocks - should now look as follows:

Program PGM01:

```
* Example Program 'PGM01' for User's Guide Tutorial
* PROGRAM NOW USES A LOCAL DATA AREA
* A GLOBAL DATA AREA AND TITLE HAVE BEEN ADDED AND
* THE DISPLAY STATEMENT HAS BEEN CHANGED
* -----
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL USING LDA01
END-DEFINE *
REPEAT
*
  INPUT USING MAP 'MAP01'
  IF #NAME-START = '.'
    ESCAPE BOTTOM
  END-IF
  MOVE #NAME-START TO #NAME-END
*
  RD1. READ EMPLOYEES-VIEW BY NAME
    STARTING FROM #NAME-START
    THRU #NAME-END
    IF LEAVE-DUE >= 20
      PERFORM MARK-SPECIAL-EMPLOYEES
    ELSE
      RESET #MARK
    END-IF
*
WRITE TITLE / '*** PERSONS WITH 20 OR MORE DAYS LEAVE DUE ***'
           / '*** ARE MARKED WITH AN ASTERISK ***' // *
  DISPLAY 23X '//N A M E' NAME
           3X '//DEPT' DEPT
           3X '//LV/DUE' LEAVE-DUE
           3X '//*' #MARK *
  END-READ
*
  IF *COUNTER (RD1.) = 0
    REINPUT 'PLEASE TRY ANOTHER NAME'
  END-IF
*
END-REPEAT
*
DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
  MOVE '*' TO #MARK
END-SUBROUTINE
END
```

Step 7

Once you have completed all changes, CHECK the program and correct any errors that may have occurred.

Then RUN the program; on the input screen, enter the name JONES.

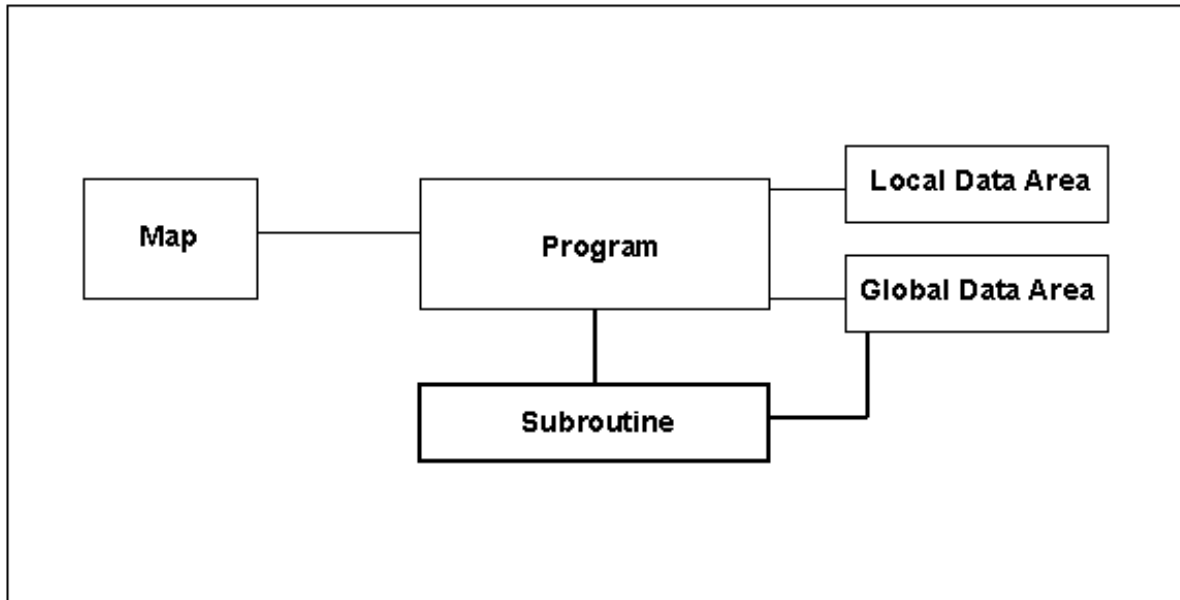
Note the differences in the report output, which should now look as follows:

*** PERSONS WITH 20 OR MORE DAYS LEAVE DUE ***			
*** ARE MARKED WITH AN ASTERISK ***			
N A M E	DEPT	LV	*
-----	-----	---	-
JONES	SALE30	25	*
JONES	MGMT10	34	*
JONES	TECH10	11	
JONES	MGMT10	18	
JONES	TECH10	21	*
JONES	SALE00	30	*
JONES	SALE20	14	
JONES	COMP12	26	*
JONES	TECH02	25	*

When the execution of the program has finished without any errors, STOW the program for future modification in Session 4.

End of Session 3.

Session 4 - Creating an External Subroutine



In Natural, a *subroutine* can be defined either within a program, or as an external subroutine outside the program.

Until now, the subroutine MARK-SPECIAL-EMPLOYEES has been defined within the program using a DEFINE SUBROUTINE statement. In this session, the subroutine will be defined as a separate object external to the program.

Because both internal and external subroutines are invoked with a PERFORM statement, only minimal changes to the program are required.

Step 1

If program PGM01 is not already in the program editor, enter the code "E" and name PGM01 on the Development Functions menu.

Make a copy of it by saving it under a different name: in the command line of the editor, enter the command SAVE SUBR01.

Enter the command READ SUBR01 to read the new copy into the work area of the editor.

Enter the command SET TYPE S to change the object type from *program* to *subroutine*.

Step 2

Delete all lines of the subroutine except the comment lines, the DEFINE DATA and DEFINE SUBROUTINE blocks, and the END statement, so that the program looks like the illustration below.

You can delete lines quickly by marking the first line of a block of lines with the line command ".X" and the last line of a block with ".Y" and then entering the command "DX-Y" in the command line to delete the specified block. Add a comment to identify this subroutine.

Subroutine SUBR01:

```
* Example Subroutine: SUBR01
* *****
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL  USING LDA01
END-DEFINE
*
DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
  MOVE '*' TO #MARK
END-SUBROUTINE
END
```

CHECK your changes and correct any errors. Then STOW the subroutine.

Step 3

Now that the subroutine is located in SUBR01, the internal subroutine must be removed from program PGM01.

In the command line of the editor, enter READ PGM01.

Step 4

Enter the command BOTTOM in the command line to move to the subroutine definition at the end of the program. Delete the following lines, containing the internal subroutine definition, from the program:

```
DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
  MOVE '*' TO #MARK
END-SUBROUTINE
```

The program should now look like the example below:

Program PGM01

```
* Example Program 'PGM01' for User's Guide Tutorial
* PROGRAM NOW USES A LOCAL DATA AREA
* A GLOBAL DATA AREA AND TITLE HAVE BEEN ADDED AND
* THE DISPLAY STATEMENT HAS BEEN CHANGED
* THE SUBROUTINE IS NOW EXTERNAL
* -----
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL  USING LDA01
END-DEFINE
*
REPEAT
*
  INPUT USING MAP 'MAP01'
  IF #NAME-START = '.'
    ESCAPE BOTTOM
  END-IF
  MOVE #NAME-START TO #NAME-END
*
  RD1. READ EMPLOYEES-VIEW BY NAME
        STARTING FROM #NAME-START
        THRU #NAME-END
*
    IF LEAVE-DUE >= 20
      PERFORM MARK-SPECIAL-EMPLOYEES
    ELSE
      RESET #MARK
    END-IF
*
  WRITE TITLE / '*** PERSONS WITH 20 OR MORE DAYS LEAVE DUE ***'
              / '*** ARE MARKED WITH AN ASTERISK ***' //
*
  DISPLAY 23X '//N A M E' NAME
           3X '//DEPT'   DEPT
           3X '//LV/DUE' LEAVE-DUE
           3X '//*'      #MARK
*
  END-READ
*
  IF *COUNTER (RD1.) = 0
    REINPUT 'PLEASE TRY ANOTHER NAME'
  END-IF
*
END-REPEAT
END
```

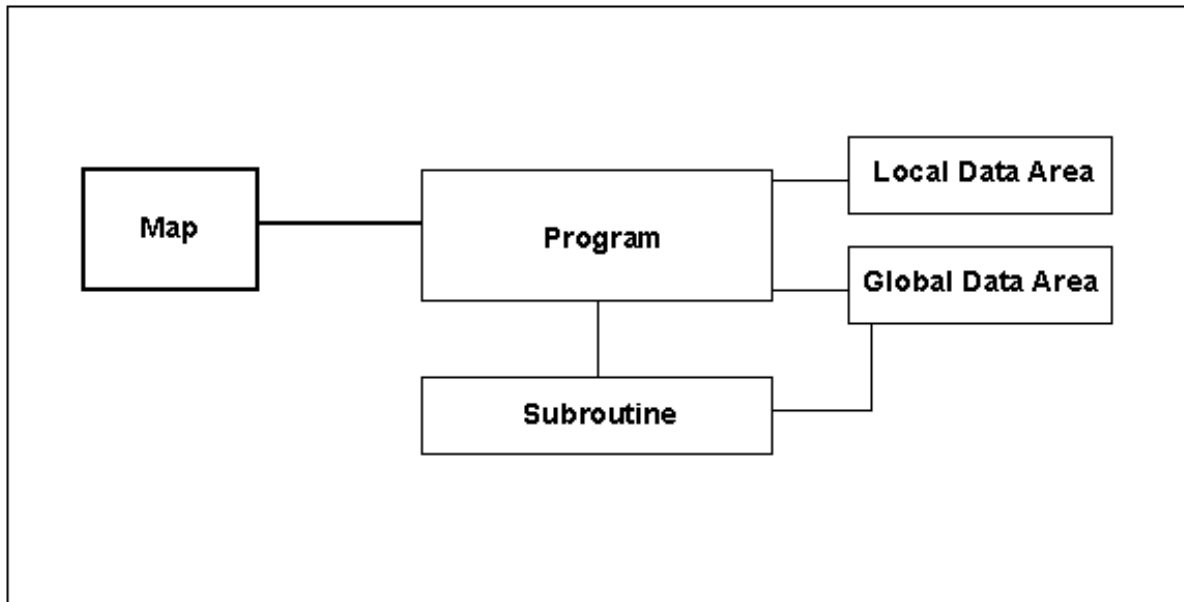
Step 5

CHECK the program and correct any errors. Then RUN the program to make sure that the results are the same with an external subroutine as previously with an internal subroutine.

STOW the program for the next session.

End of Session 4.

Session 5 - Editing a Map



In the previous sessions, the screen (map) prompting for an employee name was invoked via the INPUT USING MAP statement. In this session, the map - MAP01 - will be edited to add an ending name for a range of employees and some new layout elements.

The Natural *map editor* is used to create and modify screen layouts quickly and efficiently. In this session, you will use several map editor line commands and field commands:

- A line command begins with two periods (..). You enter it at the beginning of a line, and it applies to the whole line in which you enter it.
- A field command begins with one period (.). You enter it at the beginning of a field, and it applies only to the field in which you enter it.

(The sessions in the section Tutorial - Using the Map Editor also show you how to apply these commands to more than one line/field at a time.)

For detailed descriptions of all map editor commands, refer to Editing a Map in the section Map Editor.

Step 1

To invoke the map editor, enter the code "E" and type "M" on the Development Functions menu. The Edit Map menu will be displayed:

```

16:51:35          ***** NATURAL MAP EDITOR *****          2001-01-31
User SAG              - Edit Map -                          Library SYSTEM

      Code      Function
      ----      -
      D      Field and Variable Definitions
      E      Edit Map
      I      Initialize new Map
      H      Initialize a new Help Map
      M      Maintenance of Profiles & Devices
      S      Save Map
      T      Test Map
      W      Stow Map
      ?      Help
      .      Exit

      Code .. I      Name .. _____      Profile .. SYSPROF_

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help      Exit  Test  Edit

```

The map editor provides an extensive *help system*, which you can invoke by entering a question mark (?) as code on the Edit Map menu. Take time to look through this help system so that you know what kind of help is available.

Step 2

To edit map MAP01, enter the code "E" and name MAP01 on the Edit Map menu. The editing screen of the map editor will be displayed:

Ob	D	CLS	ATT	DEL	CLS	ATT	DEL
.		T	D	Blk	T	I	?
.		A	D	_	A	I)
.		A	N	^	M	D	&
.		M	I	:	O	D	+
.		O	I	(
.							

001 --010---+-----+-----+---030---+-----+-----+---050---+-----+-----+---070---+-----

Please enter starting name :XXXXXXXXXXXXXXXXXXXXX (. to exit)

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---

Help Mset Exit Test Edit -- - + Full < > Let

The editing screen will be displayed in split-screen mode. The top portion of the screen can be used, for example, to display view definitions; the upper right corner displays delimiter settings that apply for the map. The lower portion of the screen is the map editing area.

Map fields may be defined directly on the screen or they can be selected from a view (DDM) that is displayed in the upper portion of the screen. In this exercise, map fields will be defined directly on the screen.

Unlike the program editor and the data area editor, the map editor does not have a command line. Many functions in the map editor are performed by using PF keys (the PF-key lines at the bottom of the screen show which function is assigned to which key).

The two screens below show the map editor screen for map MAP01 as it will appear at the end of this session, and the display of the screen as it will appear when MAP01 is invoked while executing program PGM01.

Fld #NAME-END				Fmt A20			

AD= MIT'_'	ZP= OFF	SG= OFF	HE=	Rls 0			
AL=	CD=	CV=		Mod User			
PM=	DF=	DY=					
EM=							
001	--010	-----	-----030	-----	-----050	-----	-----070
(XXXXXXXX	Software AG Employee Information					(XXXXXXXX	
(XXXXXXXXXX							
Please enter starting name :XXXXXXXXXXXXXXXXXXXXX (. to exit)							
Ending name .XXXXXXXXXXXXXXXXXXXXX							
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---							
HELP Mset Exit <--- ---> -- - + < > Let							

31/01/01	Software AG Employee Information	SYSTEM
16:01:26.8		
Please enter starting name _____ (. to exit)		
Ending name _____		

Step 3

In the first line of the editing area, type in the following text, as shown below:

Software AG Employee Information

Then insert two blank lines by entering the line command `..I(2)` in the first six positions of the text you just typed in. The map now looks as follows:

```

Ob _                               Ob D CLS ATT DEL          CLS ATT DEL
.                               .      T  D   Blnk          T  I   ?
.                               .      A  D   _            A  I   )
.                               .      A  N   ^            M  D   &
.                               .      M  I   :            O  D   +
.                               .      O  I   (
.                               .
001  --010---+---+---+---030---+---+---+---050---+---+---+---070---+---
Software AG Employee Information

Please enter starting name :XXXXXXXXXXXXXXXXXXXXX (. to exit)

```

Step 4

Enter the line command `"..C"` in the first three positions of the `"Software AG Employee Information"` line. The text is now centered:

```

Ob  _                               Ob D CLS ATT  DEL      CLS ATT  DEL
.                               .      T  D      Blnk      T  I      ?
.                               .      A  D      _          A  I      )
.                               .      A  N      ^          M  D      &
.                               .      M  I      :          O  D      +
.                               .      O  I      (
.                               .
001  --010---+-----+-----030---+-----+-----050---+-----+-----070---+-----
      Software AG Employee Information

      Please enter starting name :XXXXXXXXXXXXXXXXXXXXX (. to exit)

```

Step 5

Enter (*DATE, (*TIME and (*LIBRARY-ID as shown below:

Ob	__	Ob	D	CLS	ATT	DEL	CLS	ATT	DEL
.		.		T	D	Blnk	T	I	?
.		.		A	D	_	A	I)
.		.		A	N	^	M	D	&
.		.		M	I	:	O	D	+
.		.		O	I	(
.		.							
001	--010-----+-----+-----030-----+-----+-----050-----+-----+-----070-----+-----								
(*DATE				Software AG Employee Information			(*LIBRARY-ID		
(*TIME									
Please enter starting name :XXXXXXXXXXXXXXXXXXXXX (. to exit)									

*DATE, *TIME, and *LIBRARY-ID are Natural system variables. System variables all begin with an asterisk (*). When the program which invokes the map is executed, *DATE will display the current date, *TIME the current time, and *LIBRARY-ID your current library. For more information on system variables, see the section System Variables in the Natural Programming Reference documentation.

The opening parenthesis "(" in front of the system variable is a *delimiter* character. A delimiter indicates the combination of class and attribute assigned to a field. In this case, the "(" delimiter identifies the fields as a non-modifiable, intensified, output-only fields. The currently valid delimiter characters are shown in the top right-hand corner of the map editing screen.

Class types (column CLS) shown on this screen include:

Class Type	Description
T	Text constant
A	Input field
O	Output-only field (non-modifiable)
M	Modifiable field (output and input field)

Attribute types (column ATT) shown on this screen include:

Attribute Type	Description
D	Default (that is, non-intensified, non-blinking, etc.)
I	Intensified

By entering these delimiter characters directly in front of a field, you assign a class and attribute to that field. Other class and attribute combinations are possible. It is also possible to assign another delimiter character to a specific class/attribute combination.

After you have typed in the system variables along with the delimiter characters, press ENTER. The system variable names entered will be transformed on the map to a series of Xs:

Ob _	Ob D CLS ATT DEL	CLS ATT DEL
.	. T D Blnk	T I ?
.	. A D _	A I)
.	. A N ^	M D &
.	. M I :	O D +
.	. O I (
.	.	
001 --010----	-----030----	-----050----
(XXXXXXXX	Software AG Employee Information	(XXXXXXXX
(XXXXXXXX		
Please enter starting name :XXXXXXXXXXXXXXXXXXXXX (. to exit)		

Step 6

Add another field to the map. Type in as shown in the screen below:

Ending name :X(20)

The colon (:) indicates that the field is modifiable for user input, and is displayed intensified.

Ob	__	Ob	D	CLS	ATT	DEL	CLS	ATT	DEL
.		.		T	D	Blk	T	I	?
.		.		A	D	_	A	I)
.		.		A	N	^	M	D	&
.		.		M	I	:	O	D	+
.		.		O	I	(
.		.							

001	--010	----	-----	-----	030	----	-----	-----	050	----	-----	-----	070	----	-----
(XXXXXXXX					Software AG Employee Information								(XXXXXXXX		
(XXXXXXXX															

Please enter starting name :XXXXXXXXXXXXXXXXXXXXX (. to exit)
Ending name :x(20)

Press ENTER, and the new field is added, its length determined by the 20 Xs:

Ob	__	Ob	D	CLS	ATT	DEL	CLS	ATT	DEL
.		.		T	D	Blk	T	I	?
.		.		A	D	_	A	I)
.		.		A	N	^	M	D	&
.		.		M	I	:	O	D	+
.		.		O	I	(
.		.							

001	--010	----	-----	-----	030	----	-----	-----	050	----	-----	-----	070	----	-----
(XXXXXXXX					Software AG Employee Information								(XXXXXXXX		
(XXXXXXXX															

Please enter starting name :XXXXXXXXXXXXXXXXXXXXX (. to exit)
Ending name :XXXXXXXXXXXXXXXXXXXXX

Step 7

This newly created field needs further definition before it can be processed in the program PGM01.

Enter the field command ".E" in the first two positions of the field , as shown below:

Ob	D	CLS	ATT	DEL	CLS	ATT	DEL
.		T	D	Blk	T	I	?
.		A	D	_	A	I)
.		A	N	^	M	D	&
.		M	I	:	O	D	+
.		O	I	(
.							


```

001  --010---+-----+-----030---+-----+-----050---+-----+-----070---+-----
(XXXXXXXXX      Software AG Employee Information      (XXXXXXXXX
(XXXXXXXXXXXXX

Please enter starting name :XXXXXXXXXXXXXXXXXXXXX (. to exit)
Ending name .XXXXXXXXXXXXXXXXXXXXX
  
```

This causes the extended field editing section to be invoked for the field you have marked with the command. (Another way to invoke extended field editing function is to position the cursor anywhere in the field and press PF5.)

The extended field editing section will be displayed in the top half of the screen:

Fld #001				Fmt A20

AD= MIT'_'_____	ZP= OFF	SG= OFF	HE= _____	Rls 0
AL= _____	CD= ____	CV= _____		Mod Undef
PM= ____ DF=		DY= _____		
EM= _____				

<pre> 001 --010---+-----+-----030---+-----+-----050---+-----+-----070---+----- (XXXXXXXXX Software AG Employee Information (XXXXXXXXX (XXXXXXXXXXXXX Please enter starting name :XXXXXXXXXXXXXXXXXXXXX (. to exit) Ending name .XXXXXXXXXXXXXXXXXXXXX </pre>				

The field "Fld" displays as field name a number (#001). Natural automatically assign such a number to every new field that is created in a map.

Step 8

Change the field name by typing #NAME-END over the field number. This name corresponds to the name of a user-defined variable which is defined in the local data area used by the program PGM01.

Fld	#NAME-END	Fmt	A20
AD=	MIT'_'_____	ZP=	OFF
AL=	_____	SG=	OFF
PM=	_____	HE=	_____
EM=	_____	CD=	_____
		CV=	_____
		DY=	_____

001	--010----	030----	050----
(XXXXXXXX	Software AG	Employee Information	(XXXXXXXX
(XXXXXXXX			
Please enter starting name :XXXXXXXXXXXXXXXXXXXXX (. to exit)			
Ending name .XXXXXXXXXXXXXXXXXXXXX			

The map modifications are now complete. Press PF3 to leave the extended field editing section.

Press PF3 again to return to the Edit Map menu.

Step 9

Press PF4 to test the map. You will see a display of the map as it will appear when invoked from the program PGM01:

31/01/01	Software AG Employee Information	SYSTEM
17:11:00.9		
Please enter starting name _____ (. to exit)		
Ending name _____		

Press PF3 to end testing.

Enter the code "W" on the Edit Map menu to STOW the map for future use.

Step 10

To leave the Edit Map menu, enter the code (.), and you will be returned to the Development Functions menu.

Enter the command E PGM01 in the command line. The program editor will be invoked and program PGM01 will be read into the work area ready to be adjusted to the modified map.

Step 11

Until now, PGM01 included the instruction:

```
MOVE #NAME-START TO #NAME-END
```

Thus, the start value for the list displayed by the program was also the end value, which meant that in the example used the list contained only employees whose names were JONES. (Otherwise, all employees from JONES to the end of the alphabet would have been included in the report.) Now the map allows us to specify both a start value **and** an end value for the list to be output. However, an IF statement must be added to the program to handle a situation in which no end value is specified.

Change the program by inserting the following lines before the READ statement:

```
IF #NAME-END = ' '  
    MOVE #NAME-START TO #NAME-END  
END-IF
```

Add comments to reflect program changes.

The program should now look as follows:

Program PGM01:

```
* Example Program 'PGM01' for User's Guide Tutorial
* PROGRAM NOW USES A LOCAL DATA AREA
* A GLOBAL DATA AREA AND TITLE HAVE BEEN ADDED AND
* THE DISPLAY STATEMENT HAS BEEN CHANGED
* THE SUBROUTINE IS NOW EXTERNAL
* A BEGINNING AND ENDING NAME ARE USED FOR THE OUTPUT
* -----
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL  USING LDA01
END-DEFINE
*
REPEAT
*
  INPUT USING MAP 'MAP01'
  IF #NAME-START = '.'
    ESCAPE BOTTOM
  END-IF
IF #NAME-END = ' '
  MOVE #NAME-START TO #NAME-END
END-IF*
RD1. READ EMPLOYEES-VIEW BY NAME
      STARTING FROM #NAME-START
      THRU #NAME-END
  IF LEAVE-DUE >= 20
    PERFORM MARK-SPECIAL-EMPLOYEES
  ELSE
    RESET #MARK
  END-IF
*
WRITE TITLE / '*** PERSONS WITH 20 OR MORE DAYS LEAVE DUE ***'
           / '*** ARE MARKED WITH AN ASTERISK ***' //
*
  DISPLAY 23X '//N A M E' NAME
           3X '//DEPT'   DEPT
           3X '//LV/DUE' LEAVE-DUE
           3X '//*'      #MARK
*
  END-READ
*
  IF *COUNTER (RD1.) = 0
    REINPUT 'PLEASE TRY ANOTHER NAME'
  END-IF
*
END-REPEAT
END
```

Step 12

CHECK the program and correct any errors that may be indicated.

Then RUN the program. On the input screen, enter the names JONES and JOY as start value and end value respectively. The following output report will appear:

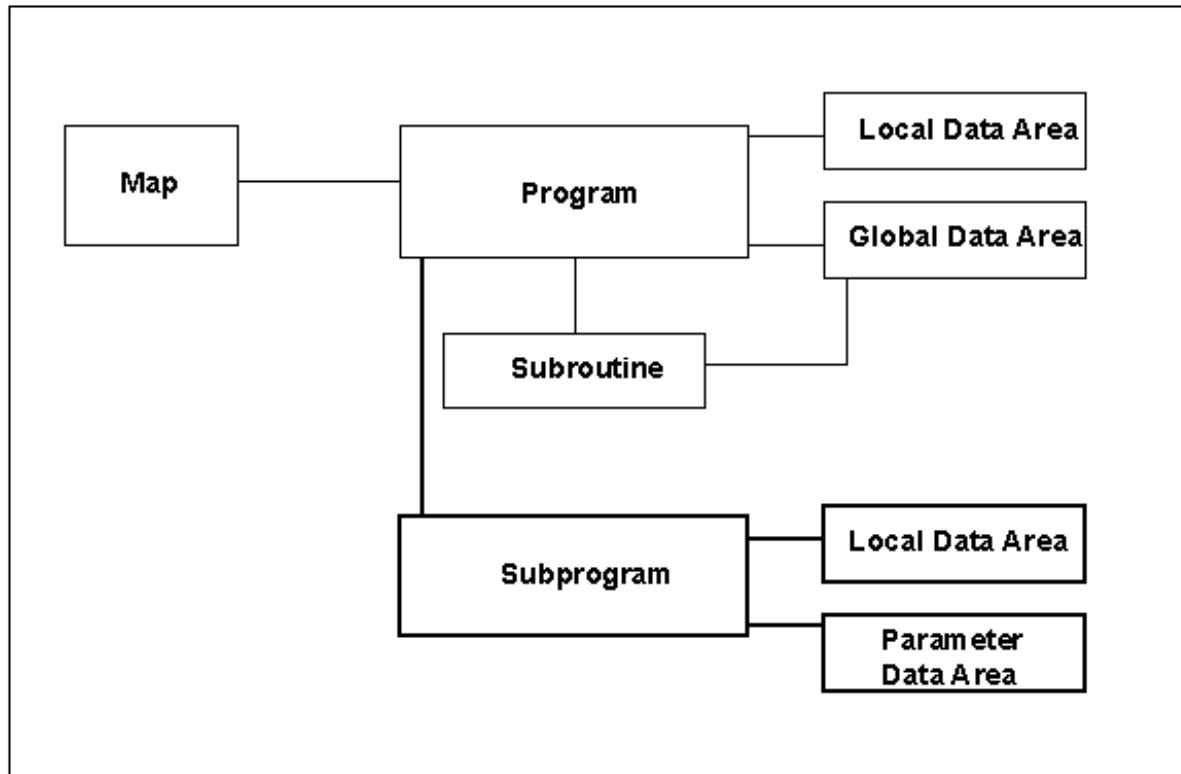
*** PERSONS WITH 20 OR MORE DAYS LEAVE DUE ***				
*** ARE MARKED WITH AN ASTERISK ***				
N A M E	DEPT	LV	DUE	*
-----	-----	---	-	
JONES	SALE30	25		*
JONES	MGMT10	34		*
JONES	TECH10	11		
JONES	MGMT10	18		
JONES	TECH10	21		*
JONES	SALE00	30		*
JONES	SALE20	14		
JONES	COMP12	26		*
JONES	TECH02	25		*
JOPER	MARK29	32		*
JOUSSELIN	FINA01	45		*

STOW the program for future use.

To return to the Development Functions menu, enter a period (.) in the command line of the editor.

End of Session 5.

Session 6 - Invoking a Subprogram



In Natural, both *subprograms* and *subroutines* can be invoked from a program. They differ from one another in the way data from the invoking program can be accessed.

As shown in Session 4, a *subroutine* can access the same global data area as the invoking program. However, a *subprogram* is invoked with a CALLNAT statement, and with this statement, parameters can be passed from the invoking program to the subprogram. These parameters are the only data available to the subprogram from the invoking program.

The passed parameters must be defined either within the DEFINE DATA PARAMETER statement of the subprogram, or in a parameter data area used by the subprogram.

In addition, a subprogram can have its own local data area, in which the fields to be used within the subprogram are defined.

In this session, you will create a subprogram with a parameter data area and a local data area. Moreover, a CALLNAT statement to invoke the subprogram will be added to the main program; also the main program's local data area has to be modified. In the subprogram, the employees selected by the main program will be the basis to select the corresponding vehicle information from the VEHICLES file. As a result, the report will contain vehicle information as well as employees information.

Step 1

First, the main program's local data area, LDA01, must be modified.

On the Development Functions menu, enter the code "E" (Edit Object) and name LDA01. The data area editor will be invoked, and the local data area LDA01 will be read into the editing area. LDA01 should appear as follows:

Local	LDA01	Library	SYSTEM	DBID	10	FNR	32
Command							> +
I	T	L	Name	F	Leng	Index/Init/EM/Name/Comment	
All	-	-	-	-	-	-	-
1			#NAME-START	A	20		
1			#NAME-END	A	20		

Step 2

Add the variables #PERS-ID, #MAKE and #MODEL to the local data area, as show below. They will be referenced in the CALLNAT statement to be added to the program. The local data area now looks as follows:

Local	LDA01	Library	SYSTEM	DBID	10	FNR	32
Command							> +
I	T	L	Name	F	Leng	Index/Init/EM/Name/Comment	
All	-	-	-	-	-	-	-
1			#NAME-START	A	20		
1			#NAME-END	A	20		
1			#PERS-ID	A	8		
1			#MAKE	A	20		
1			#MODEL	A	20		

CHECK and STOW the local data area.

Step 3

With minor modifications it is possible to use the local data area LDA01 to create the parameter data area that will be needed for the subprogram. (Instead of creating a separate parameter data area, it would also be possible to define the parameter data directly within the subprogram's DEFINE DATA PARAMETER statement.)

Make a copy of LDA01 by saving it under a different name: in the command line of the editor, enter the command SAVE PDA02.

Then enter the command READ PDA02 to read the new copy into the editor.

Then enter the command SET TYPE PARAMETER to change the data area type from Local to Parameter.

Step 4

With the line command ".D", delete the first two lines:

```
1 #NAME-START A 20
1 #NAME-END   A 20
```

The parameter data area now looks as follows:

Parameter	PDA02	Library	SYSTEM	DBID	10	FNR	32
Command							> +
I T L	Name			F	Leng	Index/Init/EM/Name/Comment	
All	-	-	-	-	-	-	-
1	#PERS-ID			A	8		
1	#MAKE			A	20		
1	#MODEL			A	20		

CHECK the parameter data area and correct any errors, and then STOW it.

Step 5

Like a program, a subprogram can have its own local data area. This will be created now. First enter the command CLEAR to clear the contents of the editing area. Then change the data area type to Local with the command SET TYPE LOCAL.

Step 6

In the first line of the editing area, enter the line command ".V (VEHICLES)" starting in column "T". The view VEHICLES will be displayed listing all fields contained in that view. Select the following fields to be included into the data area: PERSONNEL-ID, MAKE, and MODEL (see also Session 2, Step 5). These fields will be automatically incorporated into the local data area.

The new local data area should now appear as follows:

Local		Library	SYSTEM	DBID	10	FNR	32
Command							> +
I T L	Name			F	Leng	Index/Init/EM/Name/Comment	
All	-	-	-	-	-	-	-
V 1	VEHICLES-VIEW					VEHICLES	
2	PERSONNEL-ID			A	8		
2	MAKE			A	20		
2	MODEL			A	20		

Step 7

Enter the command SAVE LDA02 to store the data area in source form. Then CHECK the data area, and correct any errors if necessary. Then STOW the data area to compile it and store it in source and object form.

The local data area is now ready to be used by the subprogram.

Step 8

The next step is to create the subprogram itself.

Leave the data area editor by entering a period (.) in the command line. On the Development Functions menu, enter the code "E" (for Edit Object) and type "N" (for subprogram). The program editor will be invoked with an empty work area.

Type in the subprogram as shown below. Then SAVE it under the name SPGM02 (SAVE SPGM02). CHECK it and correct any errors, and then STOW it.

Subprogram SPGM02:

```
* Example Subprogram 'SPGM02'
* *****
DEFINE DATA
  PARAMETER USING PDA02
  LOCAL      USING LDA02
END-DEFINE
*
FD1. FIND (1) VEHICLES-VIEW WITH PERSONNEL-ID = #PERS-ID
      MOVE MAKE (FD1.)      TO #MAKE
      MOVE MODEL (FD1.)     TO #MODEL
      ESCAPE BOTTOM
END-FIND
END
```

This subprogram receives the personnel number passed by the main program and uses this number as the basis of a search of the VEHICLES file.

Step 9

Now the main program must be modified to invoke the subprogram.

Read the program into the editor with the command READ PGM01, and insert the following statements immediately before the WRITE TITLE statement:

```
RESET #MAKE #MODEL
CALLNAT 'SPGM02' PERSONNEL-ID #MAKE #MODEL
```

Some of the parameters passed in the CALLNAT statement are defined in the global data area, and some in the local data area. Note also that the variables defined in the parameter data area of the subprogram need not have the same name as the variables in the CALLNAT statement; it is only necessary that they match in sequence, format, and length.

Step 10

As the program receives vehicle information from the subprogram, the DISPLAY statement must be expanded as follows:

Previous DISPLAY Statement:

```
DISPLAY 23X ' //N A M E' NAME
          3X ' //DEPT'      DEPT
          3X ' //LV/DUE'    LEAVE-DUE
          3X ' //*'         #MARK
```

Expanded DISPLAY Statement:

```
DISPLAY   ' //N A M E' NAME
          2X ' //DEPT'      DEPT
          2X ' //LV-/DUE'   LEAVE-DUE
          ' '               #MARK
          2X ' //MAKE'      #MAKE
          2X ' //MODEL'     #MODEL
```

After you have made all modifications, the program should look as follows:

Program PGM01:

```
* Example Program 'PGM01' for User's Guide Tutorial
* PROGRAM NOW USES A LOCAL DATA AREA
* A GLOBAL DATA AREA AND TITLE HAVE BEEN ADDED AND
* THE DISPLAY STATEMENT HAS BEEN CHANGED
* THE SUBROUTINE IS NOW EXTERNAL
* A BEGINNING AND ENDING NAME ARE USED FOR THE OUTPUT
* A SUBPROGRAM PROVIDES VEHICLE INFORMATION
* -----
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL USING LDA01
END-DEFINE
*
REPEAT
*
  INPUT USING MAP 'MAP01'
  IF #NAME-START = ' .'
    ESCAPE BOTTOM
  END-IF
  IF #NAME-END = ' '
    MOVE #NAME-START TO #NAME-END
  END-IF
*
  RD1. READ EMPLOYEES-VIEW
    BY NAME
    STARTING FROM #NAME-START
    THRU #NAME-END
*
    IF LEAVE-DUE >= 20
      PERFORM MARK-SPECIAL-EMPLOYEES
    ELSE
      RESET #MARK
    END-IF
*
  RESET #MAKE #MODEL
  CALLNAT 'SPGM02' PERSONNEL-ID #MAKE #MODEL
  WRITE TITLE / '*** PERSONS WITH 20 OR MORE DAYS LEAVE DUE ***'
    / '*** ARE MARKED WITH AN ASTERISK ***'
*
  DISPLAY      '//N A M E' NAME
    2X '//DEPT'  DEPT
    2X '//LV/DUE' LEAVE-DUE
    ' '
    #MARK
    2X '//MAKE'  #MAKE
    2X '//MODEL' #MODEL
*
  END-READ
*
  IF *COUNTER (RD1.) = 0
    REINPUT 'PLEASE TRY ANOTHER NAME'
  END-IF
*
END-REPEAT
END
```

Step 11

Once the program modifications have been made, CHECK the program and correct any errors.

Then RUN the program. Enter JONES and JOY as starting and ending names on the input screen. The report produced by the program now looks as follows:

*** PERSONS WITH 20 OR MORE DAYS LEAVE DUE ***				
*** ARE MARKED WITH AN ASTERISK ***				
N A M E	DEPT	LV DUE	MAKE	MODEL
JONES	SALE30	25	CHRYSLER	* IMPERIAL
JONES	MGMT10	34	CHRYSLER	* PLYMOUTH
JONES	TECH10	11	GENERAL MOTORS	CHEVROLET
JONES	MGMT10	18	FORD	ESCORT
JONES	TECH10	21	GENERAL MOTORS	* BUICK
JONES	SALE00	30	GENERAL MOTORS	* PONTIAC
JONES	SALE20	14	GENERAL MOTORS	OLDSMOBILE
JONES	COMP12	26	DATSUN	* SUNNY
JONES	TECH02	25	FORD	* ESCORT 1.3
JOPER	MARK29	32		*
JOUSSELIN	FINA01	45	RENAULT	* R25

After the program has been executed, STOW it for future reference.

End of Session 6.

Tutorial - Using the Map Editor

This tutorial is not intended to be a comprehensive description of the full range of possibilities provided by the Natural map editor. Rather, these sessions represent a general introduction to how the map editor may be used. Therefore, explanations are kept to a minimum. For a full description of all functions and features, see the section Map Editor.

It is important that you work through the sessions in the sequence below.

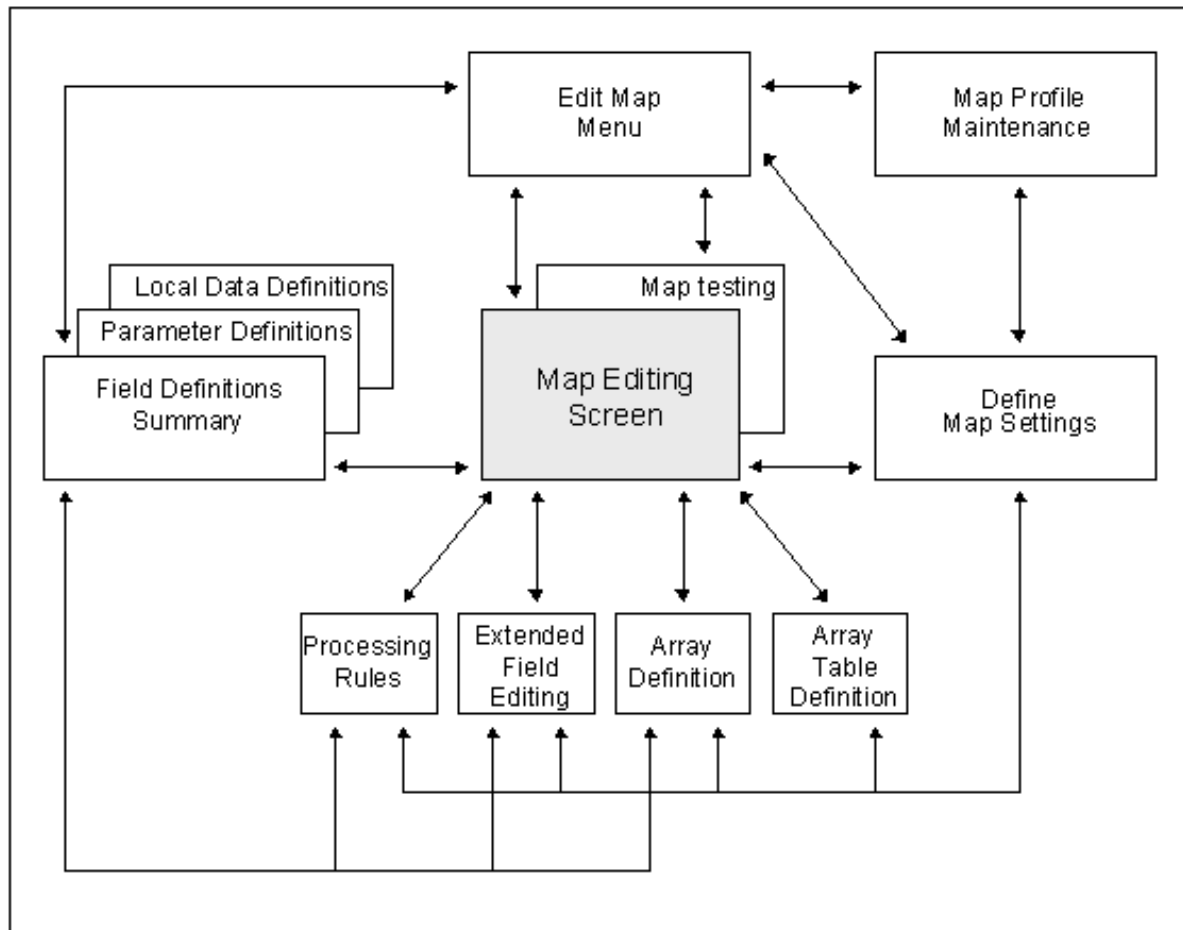
The Natural map editor is used to create *maps* (screen layouts) referenced in a program. The map editor allows *direct manipulation of fields* for screen layouts; *extended field editing* facilitates the definition of fields; *processing rules* within the map can be associated with these fields. Once a map has been created, it may be stored in the Natural system file from where it may be invoked by a Natural program using a WRITE USING MAP or INPUT USING MAP statement.

This section covers the following topics:

- Components of the Map Editor
 - Invoking the Map Editor
 - Session 1 - Designing a Map, Line and Field Commands
 - Session 2 - Processing Rules
 - Session 3 - Extended Field Editing
 - Session 4 - INPUT USING MAP
 - Session 5 - WRITE USING MAP, Fields from a View
-

Components of the Map Editor

The following diagram gives an overview of the various sections of the map editor.



Invoking the Map Editor

On the Natural Main Menu, select "Development Functions". The Development Functions menu will be displayed.

For the sessions in this tutorial, change the mode to "Structured" if you are not working in structured mode already. (If you are currently working in reporting mode, enter an "S" in the first position of the Mode field.)

Then enter an "E" (Edit) in the Code field and an "M" (Map) in the Type field. The Edit Map menu will be displayed:

```

13:40:54          ***** NATURAL MAP EDITOR *****          2001-01-31
User SAG              - Edit Map -                          Library SYSTEM

      Code      Function
      ----      -
      D      Field and Variable Definitions
      E      Edit Map
      I      Initialize new Map
      H      Initialize a new Help Map
      M      Maintenance of Profiles & Devices
      S      Save Map
      T      Test Map
      W      Stow Map
      ?      Help
      .      Exit

      Code .. I      Name .. _____      Profile .. SYSPROF_

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help      Exit  Test  Edit

```

The Edit Map menu is the main menu of the Natural map editor.

Note:

The map editor contains an extensive help system. Anytime you require help, enter a question mark (?) in the field for which you wish further information. This will invoke the online help for that field. If a field does not have an individual help assigned, a help menu will be displayed, from which you may select the desired item of information.

Session 1 - Designing a Map, Line and Field Commands

On the Edit Map menu, enter an "I" (Initialize new Map) in the Code field and MAP001 in the Name field. The Define Map Settings For Map screen will be invoked:

13:43:07				Define Map Settings for MAP				2001-01-31			
Delimiters				Format				Context			
-----				-----				-----			
Cls	Att	CD	Del	Page Size	23	Device Check	_____		
T	D		BLANK	Line Size	79	WRITE Statement		_____		
T	I		?	Column Shift	...	0 (0/1)	INPUT Statement	X	_____		
A	D		_	Layout	_____	Help	_____			
A	I)	dynamic	N (Y/N)	as field default	N (Y/N)	_____		
A	N		^	Zero Print	N (Y/N)	_____				
M	D		&	Case Default	...	UC (UC/LC)	_____				
M	I		:	Manual Skip	N (Y/N)	Automatic Rule Rank	1	_____		
O	D		+	Decimal Char	Profile Name	SYSPROF		
O	I		(Standard Keys	..	N (Y/N)	_____				
				Justification	..	L (L/R)	_____				
				Print Mode	___	_____				
				Control Var	_____	_____				
Apply changes only to new fields?				N (Y/N)	_____						
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---				_____							
Help				Quit	Let						

Move the cursor to the "Filler Characters" section of the screen. Type in an underscore (_) after each of the four options as shown below:

Justification .. L (L/R)	Filler Characters
Print Mode _	-----
Control Var _	Optional, Partial _
	Required, Partial _
Apply changes only to new fields? N (Y/N)	Optional, Complete ... _
	Required, Complete ... _
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---	
Help Quit	Let

This will cause any empty positions within an input field on the map to be filled with the underscore (_). This enables the user to see the exact position and length of a field, which makes entering input easier.

Ignore the other map settings for the time being and press ENTER. Press ENTER again. The map editing screen will be invoked:

Ob _	Ob D CLS ATT DEL	CLS ATT DEL
.	. T D Blnk	T I ?
.	. A D _	A I)
.	. A N ^	M D &
.	. M I :	O D +
.	. O I (
.	.	
001 --010---+---+---+---030---+---+---+---050---+---+---+---070---+---		
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---		
Help Mset Exit Test Edit -- - + Full << > Let		

The map editing screen will appear in split-screen format, the top half displaying the delimiter characters which are valid for the map to be created, while the bottom half is the area where you actually design a map.

In the first line of the editing area, enter the line command "..F*", and in the second line type in the text PERSONNEL INFORMATION as shown below:

```

Ob _                               Ob D CLS ATT DEL      CLS ATT DEL
.                               .   T  D   Blnk      T  I   ?
.                               .   A  D   _         A  I   )
.                               .   A  N   ^         M  D   &
.                               .   M  I   :         O  D   +
.                               .   O  I   (
.
001  --010---+-----+---030---+-----+---050---+-----+---070---+---
..F*
PERSONNEL INFORMATION

```

The result will be as follows:

```

Ob _                               Ob D CLS ATT DEL      CLS ATT DEL
.                               .   T  D   Blnk      T  I   ?
.                               .   A  D   _         A  I   )
.                               .   A  N   ^         M  D   &
.                               .   M  I   :         O  D   +
.                               .   O  I   (
.
001  --010---+-----+---030---+-----+---050---+-----+---070---+---
*****
PERSONNEL INFORMATION

```

Press PF9 to obtain the full-screen map editing area.

In the bottom line, enter the line command "..F*". The screen now appears as follows:

```

*****
PERSONNEL INFORMATION

*****
001  --010---+-----+---030---+-----+---050---+-----+---070---+---

```

Type in the line command ".C" in the first three positions of the text:

```
*****
. . CSONNEL INFORMATION
```

As a result, the text will be centered.

Enter the following as shown on the screen below:

```
*****
( *DATX                      PERSONNEL INFORMATION
( *TIMX

PLEASE ENTER CITY: :X(20)
PLEASE ENTER NAME: :XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

*DATX and *TIMX are Natural system variables which will display the current date and time respectively. The opening parenthesis "(" is the delimiter for intensified output fields. The colon (:) is the delimiter for intensified modifiable fields. The number of Xs indicate the length of the fields.

The map will appear as follows:

```
*****
( XXXXXXXX                      PERSONNEL INFORMATION
( XXXXXXXX

PLEASE ENTER CITY: :XXXXXXXXXXXXXXXXXXXX
PLEASE ENTER NAME: :XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Type in the field command ".M" and move the cursor to the position indicated by [] as shown below:

```
*****
( XXXXXXXX                      PERSONNEL INFORMATION
( XXXXXXXX

. MEASE ENTER CITY: :XXXXXXXXXXXXXXXXXXXX
PLEASE ENTER NAME: :XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
[ ]
```

When you press ENTER, the text field where the command was entered will be moved to the cursor position:

```
*****
( XXXXXXXX                      PERSONNEL INFORMATION
( XXXXXXXX

      ENTER CITY: :XXXXXXXXXXXXXXXXXXXX
PLEASE ENTER NAME: :XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE
```

Enter the line command "..M" as shown below and move the cursor to the position indicated:

```
*****
(XXXXXXXXX          PERSONNEL INFORMATION
(XXXXXXXXX

..M    ENTER CITY::XXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
[ ]LEASE
```

As a result, the line where the command was entered will be moved to the line after the one in which the cursor was positioned:

```
*****
(XXXXXXXXX          PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE
    ENTER CITY::XXXXXXXXXXXXXXXXXXXXX
```

Enter the line command "..J" as shown below:

```
*****
(XXXXXXXXX          PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
..JASE
    ENTER CITY::XXXXXXXXXXXXXXXXXXXXX
```

As a result, the line where the command was entered and the line below it will be joined:

```
*****
(XXXXXXXXX          PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXX
```

Type in some additional text in the same order and position as below:

```
*****
(XXXXXXXXX                PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXX

THIS PORTION OF TEXT IS
FOR FURTHER DEMONSTRATION
OF THE MOVE
COMMANDS
```

Press ENTER.

Now type in the field command ".M" twice as shown below to move a block of fields:

```
*****
(XXXXXXXXX                PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXX

.MIS PORTION OF TEXT IS
FOR FURTHER DEMONSTRATION
OF THE MOVE
.MMMANDS
```

[]

Move the cursor to the position indicated above. As a result, the following block of fields will be moved to the following position, the top left corner of the block being placed at the cursor position:

```

*****
(XXXXXXXXX                PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXX

        OF TEXT IS
        DEMONSTRATION

                                THIS PORTION
                                FOR FURTHER
                                OF THE MOVE
                                COMMANDS

```

The position and size of the fields where the commands are entered determine the size of the block of fields that is moved, as shown above.

Enter the field command ".M" twice as shown below and move the cursor to the position indicated:

```

*****
(XXXXXXXXX                PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXX

        .M TEXT IS
        .MMONSTRATION

                                THIS PORTION[ ]
                                FOR FURTHER
                                OF THE MOVE
                                COMMANDS

```

The field and block sizes are marked again. Note that the cursor marks the target position of the top left corner of the whole block, **not** that of the top left field within the block. The result will be the following:

```

*****
(XXXXXXXXX                PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXX

                                     THIS PORTION OF TEXT IS
                                     FOR FURTHER DEMONSTRATION
                                     OF THE MOVE
                                     COMMANDS

```

Enter the command ".M" three times to determine the entire block of fields as shown below:

```

*****
(XXXXXXXXX                PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXX

                                     [ ]

                                     .MIS PORTION OF TEXT IS
                                     FOR FURTHER .MMONSTRATION
                                     OF THE MOVE
                                     .MMMANDS

```

Move the cursor to the position indicated above.

The block of fields will be moved to the position shown below:

```
*****
(XXXXXXXXX          PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXX

                THIS PORTION OF TEXT IS
                FOR FURTHER DEMONSTRATION
                OF THE MOVE
                COMMANDS
```

Enter the line command "..M" twice as shown below:

```
*****
(XXXXXXXXX          PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXX

..M                THIS PORTION OF TEXT IS
                   FOR FURTHER DEMONSTRATION
                   OF THE MOVE
..M                COMMANDS

[ ]
```

Move the cursor to the position indicated above.

The block of lines marked above will be placed below the line in which the cursor is positioned:

```
*****
(XXXXXXXXX          PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXX

                THIS PORTION OF TEXT IS
                FOR FURTHER DEMONSTRATION
                OF THE MOVE
                COMMANDS
```


Enter the command ".T" as shown below:

```
*****
(XXXXXXXXX          PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

THIS PORTION .T TEXT IS
FOR FURTHER DEMONSTRATION
OF THE MOVE
COMMANDS
```

As a result, the rest of the line, starting from the field in which the command was entered, will be deleted:

```
*****
(XXXXXXXXX          PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

THIS PORTION
FOR FURTHER DEMONSTRATION
OF THE MOVE
COMMANDS
```

Enter the field command ".D" as shown below:

```
*****
(XXXXXXXXX          PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

THIS PORTION
FOR .DRTHER DEMONSTRATION
OF THE MOVE
COMMANDS
```

The field marked with the command will be deleted:

```
*****
(XXXXXXXXX) PERSONNEL INFORMATION
(XXXXXXXXX)

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

THIS PORTION
FOR DEMONSTRATION
OF THE MOVE
COMMANDS
```

Enter the field command ".M" as shown below; then move the cursor to the position indicated:

```
*****
(XXXXXXXXX) PERSONNEL INFORMATION
(XXXXXXXXX)

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

THIS PORTION
FOR [ ] .M MONSTRATION
OF THE MOVE
COMMANDS
```

The field marked with the command will be moved to the cursor position:

```
*****
(XXXXXXXXX) PERSONNEL INFORMATION
(XXXXXXXXX)

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

THIS PORTION
FOR DEMONSTRATION
OF THE MOVE
COMMANDS
```

Enter the line command "..D" twice as shown below to delete a block of lines:

```

*****
(XXXXXXXXX                PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

..D                        THIS PORTION
                           FOR DEMONSTRATION
                           OF THE MOVE
..D                        COMMANDS

*****
001  --010---+-----+-----+---030---+-----+-----+---050---+-----+-----+---070---+-----

```

The block of lines delimited by the commands will be deleted:

```

*****
(XXXXXXXXX                PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXX

*****

001  --010---+-----+-----+---030---+-----+-----+---050---+-----+-----+---070---+-----

```

Enter the line command "..I4" as shown below:

```

*****
(XXXXXXXXX                PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXX
..I4

*****

001  --010---+-----+-----+---030---+-----+-----+---050---+-----+-----+---070---+-----

```

Four empty lines will be inserted, and the bottom line with the asterisks will be moved four lines down.

Press PF4 to test the map:

```
*****
97-01-31                                PERSONNEL INFORMATION
14:14:21

PLEASE ENTER NAME: _____
PLEASE ENTER CITY: _____

*****
```

Press PF3 to end testing of the map. The map editing screen will appear again.

Press PF3 again to end map editing. The Field and Variable Definitions Summary screen will appear. This will be discussed in a later session.

Press ENTER. The Edit Map menu will appear with Name set to MAP001. Enter "S" in the Code field. The map is now saved in source form.

End of Session 1.

Session 2 - Processing Rules

On the Edit Map menu, enter the code "E" and name MAP001 (if it is not already entered).

The map editing screen will appear in split-screen mode with map MAP001 being read into the editing area.

Enter the command ".P" as shown below:

Ob	D	CLS	ATT	DEL	CLS	ATT	DEL
.		T	D	Blk	T	I	?
.		A	D	_	A	I)
.		A	N	^	M	D	&
.		M	I	:	O	D	+
.		O	I	(
.							


```

001  --010---+-----+-----+---030---+-----+-----+---050---+-----+-----+---070---+-----
*****
(XXXXXXXXX                PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME: .PXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY: :XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -    +    Full  <    >    Let
  
```

This will invoke the *processing rule* editor for the field in which the command was entered:

```

Variables used in current map                                     Mod
#002(A40)
#001(A20)

Rule                                                             Field #002
>                                                                > + Rank 0      S 0    L 1      Struct Mode
ALL  ....+....10...+....+....+....30...+....+....+....50...+....+....+....70..
0010
0020
0030
0040
0050
0060
0070
0080
0090
0100
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      HELP  Mset  Exit  Test      --    -    +      Full  Sc=      Let

```

Type in the following processing rule:

```

Rule                                                             Field #002
>                                                                > + Rank 0      S 0    L 1      Struct Mode
ALL  ....+....10...+....+....+....30...+....+....+....50...+....+....+....70..
0010 *
0020 IF & = ' ' REINPUT 'PLEASE TYPE IN A NAME'
0030          MARK *&
0040 END-IF
0050 *
0060
0070
0080
0090
0100
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      HELP  Mset  Exit  Test      --    -    +      Full  Sc=      Let

```

The ampersand (&) in the processing rule will be dynamically substituted by the name of the field to which the processing rule is attached.

Press ENTER. Then press PF3 to return to the map editing screen.

Then press PF4 to test the map.

Now you may also test the processing rule: press ENTER. As a result, the processing rule will be executed:

```
*****
01-01-31                      PERSONNEL INFORMATION
14:21:56

PLEASE ENTER NAME: _____
PLEASE ENTER CITY: _____

*****
PLEASE TYPE IN A NAME
```

Note:

The text PLEASE TYPE IN A NAME may not necessarily appear at the bottom of the screen (as shown above) but on another line, depending on the position of the message line as set by the Natural administrator.

Press CLEAR to end testing of the map. The map editing screen will appear again.

Enter the command ".P" in the same position as before. The processing rule for rank 0 of the field where the command was entered will be displayed again.

Enter the command "P=5" as shown below:

```

Rule                                     Field #002
> P=5                                  > + Rank 0      S 5    L 1      Struct Mode
ALL  ....+....10...+....+....+....30...+....+....+....50...+....+....+....70..
0010 *
0020 IF & = ' ' REINPUT 'PLEASE TYPE IN A NAME'
0030          MARK *&
0040 END-IF
0050 *
0060
0070
0080
0090
0100
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      HELP  Mset  Exit  Test      --    -    +      Full  Sc=      Let

```

As a result, the processing rule which previously was assigned rank 0 will now be assigned rank 5 (processing rules are processed in ascending order of rank, starting with rank 0).

Enter the command P0 as shown below:

```

Rule                                     Field #002
> P0                                   > + Rank 5      S 5    L 1      Struct Mode
ALL  ....+....10...+....+....+....30...+....+....+....50...+....+....+....70..
0010 *
0020 IF & = ' ' REINPUT 'PLEASE TYPE IN A NAME'
0030          MARK *&
0040 END-IF
0050 *
0060
0070
0080
0090
0100
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      HELP  Mset  Exit  Test      --    -    +      Split Sc=      Let

```

An empty editor screen will be displayed, because there is no longer any processing rule assigned to rank 0.

```

Rule                                     Field #002
>                                     > + Rank 0      S 0   L 1   Struct Mode
ALL  ....+....10...+....+....+....30...+....+....+....50...+....+....+....70..
0010
0020
0030
0040
0050
0060
0070
0080
0090
0100
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      HELP  Mset  Exit  Test      --    -    +    Full  Sc=      Let

```

Type in the following processing rule:

```

Rule                                     Field #002
>                                     > + Rank 0      S 0   L 1   Struct Mode
ALL  ....+....10...+....+....+....30...+....+....+....50...+....+....+....70..
0010 *
0020 IF & = MASK ( '.' ) STOP
0030 END-IF
0040 *
0050
0060
0070
0080
0090
0100
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      HELP  Mset  Exit  Test      --    -    +    Full  Sc=      Let

```

Press ENTER. Then press PF3 to return to the map editing screen.

Press PF3 again. The Field and Variable Definitions Summary screen will be invoked:

14:27:32	Field and Variable Definitions - Summary					2001-01-31
Cmd	Field Name (Truncated)	Mod	Format	Ru	Lin	Col
___	*DATX	S	D		2	2
___	*TIMX	S	T		3	2
___	#002		A40	1	5	20
___	#001		A20		6	20

The fields contained in the map are listed in the order in which they appear on the map. The two user-defined fields are preceded by a hash/number (#). In order to be able to stow the map, you must name these fields. Type in the following names as shown below:

14:28:21	Field and Variable Definitions - Summary					2001-01-31
Cmd	Field Name (Truncated)	Mod	Format	Ru	Lin	Col
___	*DATX	S	D		2	2
___	*TIMX	S	T		3	2
___	#NAME		A40	1	5	20
___	#CITY		A20		6	20

Press ENTER twice. The Edit Map menu will appear. Enter the code "W" to stow the map. The map MAP001 is now stored in source and object form.

End of Session 2.

Session 3 - Extended Field Editing

On the Edit Map menu, enter the code "E" and name MAP001 (if it is not already entered). The map editing screen will appear with map MAP001 being read into the editing area.

On the map, enter some additional text as shown below:

Ob	D	CLS	ATT	DEL	CLS	ATT	DEL
.		T	D	Blk	T	I	?
.		A	D	_	A	I)
.		A	N	^	M	D	&
.		M	I	:	O	D	+
.		O	I	(
.							


```

001  --010---+-----+-----030---+-----+-----050---+-----+-----070---+-----
*****
(XXXXXXXXX                PERSONNEL INFORMATION
(XXXXXXXXX

PLEASE ENTER NAME::XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXXXXXXXXX

TYPE IN?. TO STOP OR?? FOR HELP.

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -    +    Full  <<    >    Let
  
```

The question mark (?) is the delimiter for intensified text fields.

Then enter the command ".E" as shown below:

Ob	D	CLS	ATT	DEL	CLS	ATT	DEL
.		T	D	Blk	T	I	?
.		A	D	_	A	I)
.		A	N	^	M	D	&
.		M	I	:	O	D	+
.		O	I	(
.							

001 --010---+-----+-----030---+-----+-----050---+-----+-----070---+-----

 (XXXXXXXX PERSONNEL INFORMATION
 (XXXXXXXX

 PLEASE ENTER NAME:.eXXX
 PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXX

 TYPE IN?. TO STOP OR?? FOR HELP.

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11--PF12---
 Help Mset Exit Test Edit -- - + Full << > Let

The *extended field editing* facility for the field in which the command was entered will appear:

Fld	#NAME	Fmt	A40
AD=	MIT'_'_____	ZP=	OFF
AL=	_____	SG=	OFF
PM=	___ DF=	HE=	_____
EM=	_____	CD=	___
		CV=	_____
		DY=	_____

Rls 1
Mod User

001 --010---+-----+-----030---+-----+-----050---+-----+-----070---+-----

 (XXXXXXXX PERSONNEL INFORMATION
 (XXXXXXXX

 PLEASE ENTER NAME:.EXXX
 PLEASE ENTER CITY::XXXXXXXXXXXXXXXXXXXXX

 TYPE IN?. TO STOP OR?? FOR HELP.

In the field "Fmt" enter A20, and in the field "HE=" enter 'HELP001' (in apostrophes!) as shown below:

Fld #NAME				Fmt A20

AD= MIT'_'	ZP= OFF	SG= OFF	HE= 'HELP001'	Rls 1
AL=	CD=	CV=		Mod User
PM=	DF=	DY=		
EM=				
001 --010---+-----+-----030---+-----+-----050---+-----+-----070---+-----				

(XXXXXXXX PERSONNEL INFORMATION				
(XXXXXXXX				
PLEASE ENTER NAME:.XXXXXXXXXXXXXXXXXXXX				
PLEASE ENTER CITY: :XXXXXXXXXXXXXXXXXXXX				
TYPE IN?. TO STOP OR?? FOR HELP.				

The field length is now reduced to 20. HELP001 (which is yet to be created) is now assigned as helproutine/help map to the field.

Press PF3 to return to the map editing screen. Then press PF3 again to return to the Edit Map menu. Enter the code "W" to stow map MAP001.

On the Edit Map menu, enter the code "H" and name HELPMAP. The Define Map Settings for HELPMAP screen will be invoked.

The Page Size is set to 23, the Line Size to 79. Change the Page Size to 15 and the Line Size to 25 by typing over the existing values.

The map settings should now look as below:

14:34:20				Define Map Settings for HELPMAP		2001-01-31	
Delimiters				Format		Context	
-----				-----		-----	
Cls	Att	CD	Del	Page Size 15	Device Check _____
T	D		BLANK	Line Size 25	WRITE Statement	
T	I		?	Column Shift	... 0 (0/1)	INPUT Statement	X
A	D		_	Layout _____		00000
A	I)	dynamic N (Y/N)		N
A	N		^	Zero Print N (Y/N)	Position Line	Col
M	D		&	Case Default	... UC (UC/LC)	Automatic Rule Rank	1
M	I		:	Manual Skip N (Y/N)	Profile Name SYSPROF
O	D		+	Decimal Char	Filler Characters	
O	I		(Standard Keys	.. N (Y/N)	-----	
				Justification	.. L (L/R)	Optional, Partial
				Print Mode _	Required, Partial
				Control Var _____	Optional, Complete	...
Apply changes only to new fields?				N (Y/N)		Required, Complete	...
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---							
Help				Quit		Let	

When you press ENTER twice, the map editing screen will appear. Press PF9 to switch to full-screen mode:

[illegible]

The portion of the screen not to be used is filled with lines of periods.

Enter some text as follows:

```

Type in the name of an
employee in the first
field and press ENTER.
You will then receive
a list of all employees
of that name.

For a list of employees
of a certain name who
live in a certain city,
type in a name in the
first field and a city
in the second field
and press ENTER.

```


Then press PF3 to return to the Edit Map menu. Enter the code "W" to stow help map HELP001.

Enter the code "T" and name MAP001 to test MAP001.

Enter a question mark (?) in the first position of the field entitled PLEASE ENTER NAME. Help map HELP001 will be displayed:

```

*****
01-01-31                PERSONNEL INFORMATION
15:12:06

PLEASE ENTER NAME: ? _____
PLEASE ENTER CITY:  _____
                                +-----+
TYPE IN . TO STOP  !               !
                   ! Type in the name of an      !
                   ! employee in the first        !
                   ! field and press ENTER.       !
                   ! You will then receive        !
                   ! a list of all employees      !
                   ! of that name.                !
                   !                             !
                   ! For a list of employees      !
                   ! of a certain name who        !
                   ! live in a certain city,      !
                   ! type in a name in the        !
                   ! first field and a city       !
                   ! in the second field         !
                   ! and press ENTER.             !
*****              !               ! *****
                                +-----+

```

Press ENTER.

Press ENTER again to test the processing rule for the first field.

Press CLEAR to end testing. The Edit Map menu will appear again.

End of Session 3.

Session 4 - INPUT USING MAP

- If you have access to a copy of the program PROG001, enter EDIT PROG001 in the Command line of the Edit Map menu. (By default, sample programs are provided in the system library SYSEXPG; ask your natural administrator for details.)
PROG001 will be read into the program editor. Make sure that the program is identical to the one shown below.
- If you do not have access to a copy of PROG001, enter EDIT PROGRAM in the Command line of the Edit Map menu.
The program editor will be invoked. If necessary, CLEAR the program editor (with the command CLEAR). Then type in the following program:

PROG001:

```

** PROG001
*****
DEFINE DATA LOCAL
01 #NAME (A20)
01 #CITY (A20)
01 PERS-VIEW VIEW OF EMPLOYEES
    02 NAME
    02 FIRST-NAME
    02 CITY
END-DEFINE
*
REPEAT
*
INPUT USING MAP 'MAP001'
*
IF #CITY NE ' ' AND #NAME NE ' '
    FIND PERS-VIEW WITH NAME = #NAME AND CITY = #CITY
    IF NO RECORDS FOUND
        REINPUT 'NO ONE BY THIS NAME LIVING IN THIS CITY.'
        MARK *#CITY
    END-NOREC
    DISPLAY NOTITLE NAME FIRST-NAME CITY
    END-FIND
ELSE
    IF #NAME NE ' '
        FIND PERS-VIEW WITH NAME = #NAME
        IF NO RECORDS FOUND
            REINPUT 'PLEASE TRY ANOTHER NAME.'
        END-NOREC
        DISPLAY NOTITLE NAME FIRST-NAME CITY
        END-FIND
    END-IF
END-IF
*
END-REPEAT
END

```

CHECK the program and correct any errors. STOW the program under the name of PROG001 (If no program name is displayed in the top line of the editor, enter STOW PROG001. If the program name PROG001 is displayed, simply enter the command STOW).

Then enter the command RUN to execute the program. Map MAP001 will be displayed.

► To see if everything works as intended

1. Press ENTER without typing in anything. As a result, the message PLEASE TYPE IN A NAME will be displayed.
2. Enter a question mark (?) in the first input field of the map. As a result, the help map "Type in the name of ... etc." will appear as a window on the map. - Press ENTER.
3. Enter the name MCKENNA in the first input field of the map. As a result, the message PLEASE TRY ANOTHER NAME will be displayed.
4. Enter the name JONES in the first input field of the map. As a result, the program will display the following list:

NAME	FIRST-NAME	CITY
-----	-----	-----
JONES	VIRGINIA	TULSA
JONES	MARSHA	MOBILE
JONES	ROBERT	MILWAUKEE
JONES	LILLY	BEVERLEY HILLS
JONES	EDWARD	CAMDEN
JONES	MARTHA	KALAMAZOO
JONES	LAUREL	BALTIMORE
JONES	KEVIN	DERBY
JONES	GREGORY	NOTTINGHAM

Keep pressing ENTER until you return to the map.

5. Enter the name JONES in the first input field and the name of the city DUNFERMLINE in the second input field. As a result, the message NO ONE BY THIS NAME LIVING IN THIS CITY will be displayed.
6. Enter the name JONES in the first input field and the name of the town TULSA in the second input field. As a result, the program will display the following list:

NAME	FIRST-NAME	CITY
-----	-----	-----
JONES	VIRGINIA	TULSA

Press ENTER to return to the map.

7. Enter a period (.) in the first input field. The program will be terminated, and you will be returned to the program editor.

End of Session 4.

Session 5 - WRITE USING MAP, Fields from a View

Enter the command SAVE PROG002 to save a copy of program PROG001 under the new name of PROG002.

Then enter the command READ PROG002 to read the newly created program PROG002 into the work area.

Modify the program to match with the one on the next page.

PROG002:

```
** PROG002
*****
DEFINE DATA LOCAL
01 #NAME (A20)
01 #CITY (A20)
01 PERS-VIEW VIEW OF EMPLOYEES
    02 NAME
    02 FIRST-NAME
    02 CITY
END-DEFINE
*
REPEAT
*
INPUT USING MAP 'MAP001'
*
IF #CITY NE ' ' AND #NAME NE ' '
    FIND PERS-VIEW WITH NAME = #NAME AND CITY = #CITY
    IF NO RECORDS FOUND
        REINPUT 'NO-ONE BY THIS NAME LIVING IN THIS CITY.'
        MARK *#CITY
    END-NOREC
*
    AT START OF DATA
        WRITE 'THE FOLLOWING EMPLOYEES LIVE IN' CITY
    END-START
    WRITE USING MAP 'MAP003'
*
    END-FIND
ELSE
    IF #NAME NE ' '
        FIND PERS-VIEW WITH NAME = #NAME
        IF NO RECORDS FOUND
            REINPUT 'PLEASE TRY ANOTHER NAME.'
        END-NOREC
*
        WRITE USING MAP 'MAP002'
*
        END-FIND
    END-IF
END-IF
*
END-REPEAT
END
```

When you have made all changes, enter the command SAVE in the command line of the program editor to save PROG002.

In the command line of the program editor, enter the command EDIT MAP.

The Edit Map menu will be displayed. Enter the code "I" and name MAP002.

The Define Map Settings For Map screen will be displayed. Change the Page Size to 60. Then type in an "X" after "WRITE Statement" and type a blank over the "X" after "INPUT Statement".

When you press ENTER, the map editing screen will be displayed. In the top line of the screen, enter "V EMPLOYEES". The fields definitions of the view (DDM) EMPLOYEES will be listed:

Ob	V	EMPLOYEES	Ob	D	CLS	ATT	DEL	CLS	ATT	DEL
1	PERSONNEL-ID	A8	.	T	D	Blk	T	I	?	
.	FULL-NAME	*G1	.							
2	FIRST-NAME	A20	.							
3	MIDDLE-I	A1	.				O	D	+	
4	NAME	A20	.	O	I	(
5	MIDDLE-NAME	A20	.							
001	--010--	-----	-----	030--	-----	-----	050--	-----	-----	070--
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--- Help Mset Exit Test Edit -- - + Full << > Let										

In the editing area, enter the following:

Ob	V	EMPLOYEES	Ob	D	CLS	ATT	DEL	CLS	ATT	DEL
1	PERSONNEL-ID	A8	.	T	D	Blk	T	I	?	
.	FULL-NAME	*G1	.							
2	FIRST-NAME	A20	.							
3	MIDDLE-I	A1	.				O	D	+	
4	NAME	A20	.	O	I	(
5	MIDDLE-NAME	A20	.							
001	--010--	-----	-----	030--	-----	-----	050--	-----	-----	070--
NAME: (4										
(2										

Two fields are now defined on the map using the field definitions of the fields NAME and FIRST-NAME taken from the view. When you press ENTER, the screen will look as below:

Ob	V	EMPLOYEES		Ob	D	CLS	ATT	DEL		CLS	ATT	DEL	
1	PERSONNEL-ID	A8	.		T	D		Blk		T	I	?	
.	FULL-NAME	*G1	.										
2	FIRST-NAME	A20	.										
3	MIDDLE-I	A1	.							O	D	+	
4	NAME	A20	.		O	I		(
5	MIDDLE-NAME	A20	.										
001	--010	----	-----	----	030	----	-----	----	050	----	-----	070	----
NAME: (XXXXXXXXXXXXXXXXXXXXX													
(XXXXXXXXXXXXXXXXXXXXX													

On the top line of the screen, type in a plus (+) sign over the "V". Repeat this step until the field "2 CITY" appears in the list. Use the minus (-) sign if you want to scroll backward.

Use the command ".M" to move the field from the second line of the editing area to the position shown below. Then enter "CITY:(2" as shown below:

Ob	V	EMPLOYEES		Ob	D	CLS	ATT	DEL		CLS	ATT	DEL	
.	FULL-ADDRESS	*G1	.		T	D		Blk		T	I	?	
1	ADDRESS-LINE	A20	.		A	D		_		A	I)	
2	CITY	A20	.		A	N		^		M	D	&	
3	ZIP	A10	.		M	I		:		O	D	+	
4	POST-CODE	A10	.		O	I		(
5	COUNTRY	A3	.										
001	--010	----	-----	----	030	----	-----	----	050	----	-----	070	----
NAME: (XXXXXXXXXXXXXXXXXXXXX (XXXXXXXXXXXXXXXXXXXXX CITY: (2													

The map should look as follows:

Ob	V	EMPLOYEES		Ob	D	CLS	ATT	DEL		CLS	ATT	DEL	
.	FULL-ADDRESS	*G1	.		T	D		Blk		T	I	?	
1	ADDRESS-LINE	A20	.		A	D		_		A	I)	
2	CITY	A20	.		A	N		^		M	D	&	
3	ZIP	A10	.		M	I		:		O	D	+	
4	POST-CODE	A10	.		O	I		(
5	COUNTRY	A3	.										
001	--010	----	-----	----	030	----	-----	----	050	----	-----	070	----
NAME: (XXXXXXXXXXXXXXXXXXXXX (XXXXXXXXXXXXXXXXXXXXX CITY: (XXXXXXXXXXXXXXXXXXXXX													

Press PF3 to return to the Edit Map menu.

Enter the code "W" to stow map MAP002.

Enter the code "I" and name MAP003. The Define Map Settings For Map screen will be displayed. Change the Page Size to 60; mark "WRITE Statement" with an "X"; unmark "INPUT Statement"; and type in MAP002 after "Layout". The map settings should be as follows:

15:33:53				Define Map Settings for MAP		2001-01-31	
Delimiters				Format		Context	
-----				-----		-----	
Cls	Att	CD	Del	Page Size 60	Device Check
T	D		BLANK	Line Size 79	WRITE Statement	X
T	I		?	Column Shift	... 0 (0/1)	INPUT Statement	
A	D		_	Layout MAP002__	Help	
A	I)	dynamic N (Y/N)	as field default	N (Y/N)
A	N		^	Zero Print N (Y/N)		
M	D		&	Case Default	... UC (UC/LC)		
M	I		:	Manual Skip N (Y/N)	Automatic Rule Rank	1
O	D		+	Decimal Char	Profile Name SYSPROF
O	I		(Standard Keys	.. N (Y/N)		
				Justification	.. L (L/R)	Filler Characters	
				Print Mode _	-----	
				Control Var	Optional, Partial
						Required, Partial
						Optional, Complete	...
						Required, Complete	...
Apply changes only to new fields?				N (Y/N)			
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---							
Help				Quit		Let	

When you press ENTER, the map editing screen will be displayed with the layout of map MAP002 in the edit area.

Use the command ".T" to delete CITY:(XXXXXXXXXXXXXXXXXXXXX.

Use the command ".M" to move the second of the remaining output fields to the right.

Insert the text "FIRST NAME:" into the line.

The map should now look as shown below:

```

Ob _                               Ob D CLS ATT  DEL      CLS ATT  DEL
.                                  .      T  D      Blnk      T  I      ?
.                                  .
.                                  .
.                                  .
.                                  .      O  D      +
.                                  .      O  I      (
.                                  .

001  --010---+---+---+---030---+---+---+---050---+---+---+---070---+---+
NAME:(XXXXXXXXXXXXXXXXXXXXX FIRST NAME: (XXXXXXXXXXXXXXXXXXXXX

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --    -      +      Full  <<      >      Let

```

Press PF3 to return to the Edit Map menu.

Enter the code "W" to stow map MAP003.

In the Command line, enter EDIT PROG002.

The program editor will appear with program PROG002 in the work area.

RUN the program. Map MAP001 will be displayed.

Enter the name JONES and no city. The list produced by the program will now use MAP002:

Page	1		2001-01-31 15:38:11
NAME: JONES	VIRGINIA	CITY: TULSA	
NAME: JONES	MARSHA	CITY: MOBILE	
NAME: JONES	ROBERT	CITY: MILWAUKEE	
NAME: JONES	LILLY	CITY: BEVERLEY HILLS	
NAME: JONES	EDWARD	CITY: CAMDEN	
NAME: JONES	MARTHA	CITY: KALAMAZOO	
NAME: JONES	LAUREL	CITY: BALTIMORE	
NAME: JONES	KEVIN	CITY: DERBY	
NAME: JONES	GREGORY	CITY: NOTTINGHAM	

Press ENTER to return to MAP001.

Enter the name JONES and the city DERBY. Map MAP003 will be displayed:

Page	2		2001-01-31 15:39:11
THE FOLLOWING EMPLOYEES LIVE IN DERBY			
NAME: JONES	FIRST NAME: KEVIN		

Press ENTER again to return to MAP001.

Enter a period (.) in the NAME field to return to the program. STOW the program.

End of Session 5.

Designing User Interfaces - Overview

The user interface of an application, that is, the way an application presents itself to the user, is a key consideration when writing an application.

This section provides information on the various possibilities Natural offers for designing user interfaces that are uniform in presentation and provide powerful mechanisms for user guidance and interaction.

When designing user interfaces, standards and standardization are key factors.

Using Natural, you can offer the end user common functionality across various hardware and operating systems.

This includes the general screen layout (information, data and message areas), function-key assignment and the layout of windows.

This section covers the following topics:

- **Screen Design**
Defining the general layout of screens.
- **Dialog Design**
Designing user interfaces.

Screen Design

This section provides options to define a general screen layout:

- Control of Function-Key Lines - Terminal Command %Y
 - Control of the Message Line - Terminal Command %M
 - Assigning Colors to Fields - Terminal Command %=
 - Outlining - Terminal Command %D=B
 - Statistics Line/Infoline - Terminal Command %X
 - Windows
 - Standard/Dynamic Layout Maps
 - Multilingual User Interfaces
 - Skill-Sensitive User Interfaces
-

Control of Function-Key Lines - Terminal Command %Y

With the terminal command %Y you can define how and where the Natural function-key lines are to be displayed.

Below is information on:

- Format of Function-Key Lines
- Positioning of Function-Key Lines
- Cursor-Sensitivity

Format of Function-Key Lines

The following terminal commands are available for defining the format of function-key lines:

%YN

The function-key lines are displayed in tabular Software AG format:

```
Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help      Exit                                Canc
```

%YS

The function-key lines display the keys sequentially and only show those keys to which names have been assigned (PF1=*value*,PF2=*value*,etc.):

```
Command ==>
PF1=Help,PF3=Exit,PF12=Canc
```

%YP

The function-key lines are displayed in PC-like format, that is, sequentially and only showing those keys to which names have been assigned (F1=*value*,F2=*value*,etc.):

```
Command ==>
F1=Help,F3=Exit,F12=Canc
```

Other Display Options

Various other command options are available for function-key lines, such as:

- single- and double-line display,
- intensified display,
- reverse video display,
- color display.

For details on these options, see %Y - Control of PF-Key Lines in the section Terminal Commands in the Natural Programming Reference documentation.

Positioning of Function-Key Lines

%YB

The function-key lines are displayed at the bottom of the screen:

16:50:53	***** NATURAL *****	2001-01-30
User SAG	- Main Menu -	Library XYZ
Function		
_ Development Functions		
_ Development Environment Settings		
_ Maintenance and Transfer Utilities		
_ Debugging and Monitoring Utilities		
_ Example Libraries		
_ Other Products		
_ Help		
_ Exit NATURAL Session		
Command ==>		
Enter-PF1----	PF2----PF3----	PF4----PF5----
PF6----	PF7----	PF8----
PF9----	PF10----	PF11----
PF12----		
Help	Exit	Canc

%YT

The function-key lines are displayed at the top of the screen:

```
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help      Exit      Canc
16:50:53      ***** NATURAL *****      2001-01-30
User SAG      - Main Menu -      Library XYZ

      Function
      _ Development Functions
      _ Development Environment Settings
      _ Maintenance and Transfer Utilities
      _ Debugging and Monitoring Utilities
      _ Example Libraries
      _ Other Products
      _ Help
      _ Exit NATURAL Session

Command ==>
```

%Ynn

The function-key lines are displayed on line *nn* of the screen. In the example below the function-key line has been set to line 10:

```

16:50:53                ***** NATURAL *****                2001-01-30
User SAG                - Main Menu -                Library XYZ

                        Function

                        _ Development Functions
                        _ Development Environment Settings
                        _ Maintenance and Transfer Utilities
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help          Exit
                        - Debugging and Monitoring Utilities
                        _ Example Libraries
                        _ Other Products
                        _ Help
                        _ Exit NATURAL Session

Command ===>

```

Cursor-Sensitivity**%YC**

This command makes the function-key lines cursor-sensitive. This means that they act like an action bar on a PC screen: you just move the cursor to the desired function-key number or name and press ENTER, and Natural reacts as if the corresponding function key had been pressed.

To switch cursor-sensitivity off, you enter %YC again (toggle switch).

By using %YC in conjunction with tabular display format (%YN) and having only the function-key names displayed (%YH), you can equip your applications with very comfortable action bar processing: the user merely has to select a function name with the cursor and press ENTER, and the function is executed.

Control of the Message Line - Terminal Command %M

Various options of the terminal command %M are available for defining how and where the Natural message line is to be displayed.

Below is information on:

- Positioning the Message Line
- Message Line Protection
- Message Line Color

Positioning the Message Line

%MB

The message line is displayed at the bottom of the screen:

```
16:50:53                ***** NATURAL *****                2001-01-30
User SAG                  - Main Menu -                          Library XYZ

                                Function

                                _ Development Functions
                                _ Development Environment Settings
                                _ Maintenance and Transfer Utilities
                                _ Debugging and Monitoring Utilities
                                _ Example Libraries
                                _ Other Products
                                _ Help
                                _ Exit NATURAL Session

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help      Exit                                Canc
Please enter a function.
```


%MT

The message line is displayed at the top of the screen:

```

Please enter a function.
16:50:53          ***** NATURAL *****          2001-01-30
User SAG          - Main Menu -          Library XYZ

          Function

          _ Development Functions
          _ Development Environment Settings
          _ Maintenance and Transfer Utilities
          _ Debugging and Monitoring Utilities
          _ Example Libraries
          _ Other Products
          _ Help
          _ Exit NATURAL Session

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
          Help          Exit                                Canc

```

Other options for the positioning of the message line are described in %M - Control of Message Line in the section Terminal Commands in the Natural Programming Reference documentation.

Message Line Protection

%MP

The message line is switched from unprotected to protected mode or vice versa. In unprotected mode, the message line can also be used for terminal input.

Message Line Color

%M=*color-code*

The message line is displayed in the specified color (for an explanation of color codes, see the session parameter CD as described in the Natural Parameter Reference documentation).

Assigning Colors to Fields - Terminal Command %=

You can use the terminal command %= to assign colors to field attributes for programs that were originally not written for color support. The command causes all fields/text defined with the specified attributes to be displayed in the specified color.

If predefined color assignments are not suitable for your terminal type, you can use this command to override the original assignments with new ones.

You can also use the %= terminal command within Natural editors, for example to define color assignments dynamically during map creation.

Codes	Description
<i>blank</i>	Clear color translate table.
F	Newly defined colors are to override colors assigned by the program.
N	Color attributes assigned by program are not to be modified.
O	Output field.
M	Modifiable field (output and input).
T	Text constant.
B	Blinking
C	Italic
D	Default
I	Intensified
U	Underlined
V	Reverse video
BG	Background
BL	Blue
GR	Green
NE	Neutral
PI	Pink
RE	Red
TU	Turquoise
YE	Yellow

Example:

`%=TI=RE,OB=YE`

This example assigns the color red to all intensified text fields and yellow to all blinking output fields.

Outlining - Terminal Command %D=B

Outlining (boxing) is the capability to generate a line around certain fields when they are displayed on the terminal screen. Drawing such "boxes" around fields is another method of showing the user the lengths of fields and their positions on the screen.

Outlining is only available on certain types of terminals, usually those which also support the display of double-byte character sets.

The terminal command %D=B is used to control outlining. For details on this command, see the relevant section in Terminal Commands in the Natural Programming Reference documentation.

Statistics Line/Infoline - Terminal Command %X

This terminal command controls the display of the Natural statistics line/infoline. The line can be used either as a statistics line or as an infoline, but not both at the same time.

Below is information on:

- Statistics Line
- Infoline

Statistics Line

To turn the statistics line on/off, enter the terminal command %X (this is a toggle function). If you set the statistics line on, you can see statistical information, such as:

- the number of bytes transmitted to the screen during the previous screen operation,
- the logical line size of the current page,
- the physical line size of the window.

For full details regarding the statistics line, see the terminal command %X as described in the Natural Programming Reference documentation.

The example below shows the statistics line displayed at the bottom of the screen:

```
>
All      ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
0010 SET CONTROL 'XT'
0020 SET CONTROL 'XI+'
0030 DEFINE PRINTER (2) OUTPUT 'INFOLINE'
0040 WRITE (2) 'EXECUTING' *PROGRAM 'BY' *INIT-USER
0050 WRITE 'TEST OUTPUT'
0070 END
0080
0090
0100
0110
0120
0130
0140
0150
0160
0170
0180
0190
0200
IO=264,AI =292,L=0 C= ,LS=80,P =23,PLS=80,PCS=24,FLD=82,CLS=1,ADA=0
```

Infoline

You can also use the statistics line as an *infoline* where status information can be displayed, for example, for debugging purposes, or you can use it as a separator line (as defined by SAA standards).

To define the statistics line as an infoline, you use the terminal command %XI+.

Once you have activated the infoline with the above command, you can define the infoline as the output destination for data with the DEFINE PRINTER statement as demonstrated in the example below:

Example:

```
SET CONTROL 'XT'  
SET CONTROL 'XI+'  
DEFINE PRINTER (2) OUTPUT 'INFOLINE'  
WRITE (2) 'EXECUTING' *PROGRAM 'BY' *INIT-USER  
WRITE 'TEST OUTPUT'  
END
```

When the above program is run, the status information is displayed in the infoline at the top of the output display:

EXECUTING POS	BY SAG	
Page 1		2001-01-22 10:56:06
TEST OUTPUT		

For further details on the statistics line/infoline, see the terminal command %X as described in the Natural Programming Reference documentation.

Windows

Below is information on:

- What is a Window?
- DEFINE WINDOW Statement
- INPUT WINDOW Statement

What is a Window?

A *window* is that segment of a logical page, built by a program, which is displayed on the terminal screen.

A *logical page* is the output area for Natural; in other words the logical page contains the current report/map produced by the Natural program for display. This logical page may be larger than the physical screen.

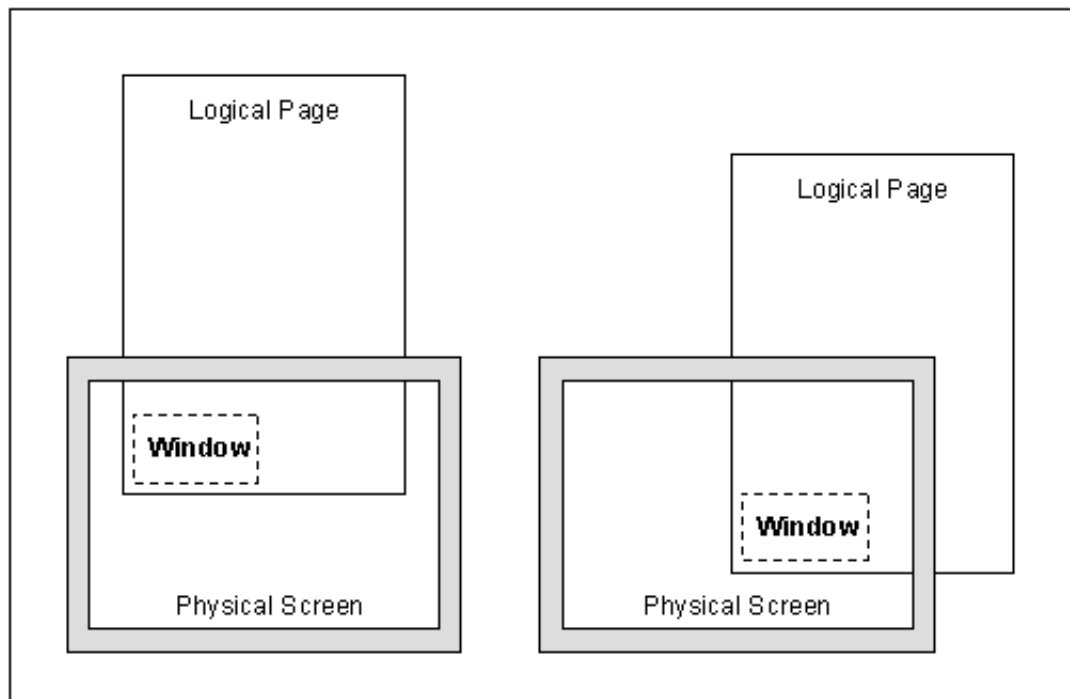
There is always a window present, although you may not be aware of its existence. Unless specified differently (by a DEFINE WINDOW statement), the size of the window is identical to the physical size of your terminal screen.

You can manipulate a window in two ways:

- You can control the size and position of the window on the *physical screen*.
- You can control the position of the window on the *logical page*.

Positioning on the Physical Screen

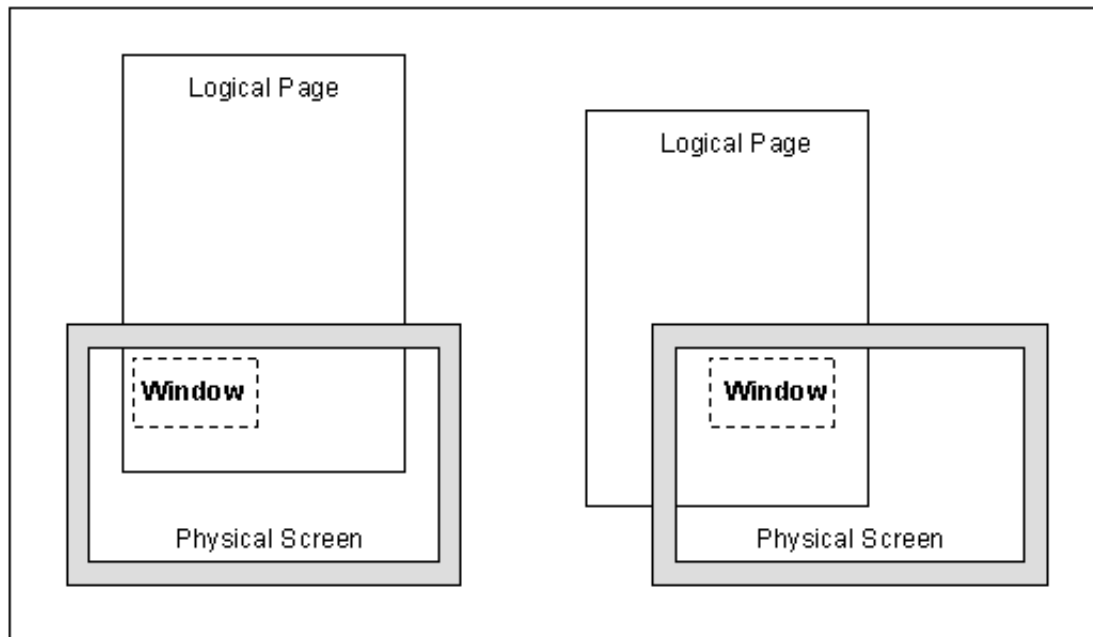
The figure below illustrates the positioning of a window on the physical screen. Note that the same section of the logical page is displayed in both cases, only the position of the window on the screen has changed.



Positioning on the Logical Page

The figure below illustrates the positioning of a window on the logical page.

When you change the position of the window on the *logical page*, the size and position of the window on the *physical screen* will remain unchanged. In other words, the window is not moved over the page, but the page is moved "underneath" the window.



DEFINE WINDOW Statement

You use the DEFINE WINDOW statement to specify the size, position and attributes of a window on the *physical screen*.

A DEFINE WINDOW statement does not activate a window; this is done with a SET WINDOW statement or with the WINDOW clause of an INPUT statement.

Various options are available with the DEFINE WINDOW statement. These are described below in the context of the example.

The following program defines a window on the physical screen.

Example:

```
DEFINE DATA LOCAL
1 COMMAND (A10)
END-DEFINE
*
DEFINE WINDOW TEST
  SIZE 5*25
  BASE 5/40
  TITLE 'Sample Window'
  CONTROL WINDOW
  FRAMED POSITION SYMBOL BOT LEFT
INPUT WINDOW='TEST'
  WITH TEXT 'message line'
  COMMAND (AD=I) /
  'dataline 1' /
  'dataline 2' /
  'dataline 3' 'long data line'
IF COMMAND = 'TEST2'
  FETCH 'TWIND2'
ELSE
  REINPUT 'invalid command'
END-IF
END
```

The *window-name* identifies the window. The name may be up to 32 characters long. For a window name, the same naming conventions apply as for user-defined variables. Here the name is TEST.

The window size is set with the SIZE option. Here the window is 5 lines high and 25 columns (positions) wide.

The position of the window is set by the BASE option. Here the top left-hand corner of the window is positioned on line 5, column 40.

With the TITLE option, you can define a title that is to be displayed in the window frame (of course, only if you have defined a frame for the window).

With the FRAMED option, you define that the window is to be framed.

This frame is then cursor-sensitive. Where applicable, you can page forward, backward, left or right within the window by simply placing the cursor over the appropriate symbol (<, -, +, or >; see POSITION clause below) and then pressing ENTER. In other words, you are moving the *logical page* underneath the window on the physical screen. If no symbols are displayed, you can page backward and forward within the window by placing the cursor in the top frame line (for backward positioning) or bottom frame line (for forward positioning) and then pressing ENTER.

With the POSITION clause of the FRAME option, you define that information on the position of the window on the logical page is to be displayed in the frame of the window. This applies only if the logical page is larger than the window; if it is not, the POSITION clause will be ignored. The position information indicates in which directions the logical page extends above, below, to the left and to the right of the current window.

If the POSITION clause is omitted, POSITION SYMBOL TOP RIGHT applies by default.

POSITION SYMBOL causes the position information to be displayed in form of symbols: "More: < - + >". The information is displayed in the top and/or bottom frame line.

TOP/BOTTOM determines whether the position information is displayed in the top or bottom frame line.

LEFT/RIGHT determines whether the position information is displayed in the left or right part of the frame line.

You can define which characters are to be used for the frame with the terminal command %F=*chv*.

<i>c</i>	The first character will be used for the four <i>corners</i> of the window frame.
<i>h</i>	The second character will be used for the <i>horizontal</i> frame lines.
<i>v</i>	The third character will be used for the <i>vertical</i> frame lines.

Example:

%F=+-!

The above command makes the window frame look like this:

```
+-----+
!                                     !
!                                     !
!                                     !
!                                     !
+-----+
```

INPUT WINDOW Statement

The INPUT WINDOW statement activates the window defined in the DEFINE WINDOW statement. In the example, the window TEST is activated. Note that if you wish to output data in a window (for example, with a WRITE statement), you use the SET WINDOW statement.

When the above program is run, the window is displayed with one input field COMMAND:

```

> r                                     > +   Program      TWIND      Lib SAG
Bot  ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
0030 END-DEFINE
0040 *                                +----Sample Window-----+
0050 DEFINE WINDOW TEST              ! message line          !
0060   SIZE 5*25                     ! COMMAND              !
0070   BASE 5/40                     ! dataline 1            !
0080   TITLE 'Sample Window'         +More:      + >-----+
0090   CONTROL WINDOW
0100   FRAMED POSITION SYMBOL BOT LEFT
0110 INPUT WINDOW='TEST'
0120   WITH TEXT 'message line'
0130   COMMAND (AD=I) /
0140   'dataline 1' /
0150   'dataline 2' /
0160   'dataline 3' 'long data line'
0170 IF COMMAND = 'TEST2'
0180   FETCH 'TWIND2'
0190 ELSE
0200   REINPUT 'invalid command'
0210 END-IF
0220 END
      ....+....1....+....2....+....3....+....4....+....5....+... S 22   L 3

```

In the bottom frame line, the position information "More + >" indicates that there is more information on the logical page than is displayed in the window.

To see the information that is further down on the logical page, you place the cursor in the bottom frame line on the plus (+) sign and press ENTER.

The window is now moved downwards. Note that the text "long data line" does not fit in the window and is consequently not fully visible.

```

> r                                     > + Program      TWIND      Lib SAG
Bot  ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
0030 END-DEFINE
0040 *                                +----Sample Window-----+
0050 DEFINE WINDOW TEST              ! invalid command      !
0060 SIZE 5*25                       ! dataline 2          !
0070 BASE 5/40                       ! dataline 3 long data !
0080 TITLE 'Sample Window'           +More:  -  >-----+
0090 CONTROL WINDOW
0100 FRAMED POSITION SYMBOL BOT LEFT
0110 INPUT WINDOW='TEST'
0120 WITH TEXT 'message line'
0130 COMMAND (AD=I) /
0140 'dataline 1' /
0150 'dataline 2' /
0160 'dataline 3' 'long data line'
0170 IF COMMAND = 'TEST2'
0180 FETCH 'TWIND2'
0190 ELSE
0200 REINPUT 'invalid command'
0210 END-IF
0220 END
      ....+....1....+....2....+....3....+....4....+....5....+... S 22  L 3

```

To see this hidden information to the right, you place the cursor in the bottom frame line on the ">" symbol and press ENTER. The window is now moved to the right on the logical page and displays the previously invisible word "line":

```

> r                                     > + Program      TWIND      Lib SAG
Bot  ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
0030 END-DEFINE
0040 *                                +----Sample Window-----+
0050 DEFINE WINDOW TEST              ! invalid command      !
0060 SIZE 5*25                       !                      !
0070 BASE 5/40                       ! line                 !
0080 TITLE 'Sample Window'           +More: < -  -----+
0090 CONTROL WINDOW
0100 FRAMED POSITION SYMBOL BOT LEFT

```

Message and Function-Key Lines

With the CONTROL clause, you determine whether the function-key lines, the message line and the statistics line are displayed in the window or on the full physical screen.

- CONTROL WINDOW displays the lines inside the window.
- CONTROL SCREEN displays the lines on the full physical screen outside the window.

If you omit the CONTROL clause, CONTROL WINDOW applies by default.

Multiple Windows

You can, of course, open multiple windows. However, only one Natural window is active at any one time, that is, the most recent window. Any previous windows may still be visible on the screen, but are no longer active and are ignored by Natural. You may enter input only in the most recent window. If there is not enough space to enter input, the window size must be adjusted first.



When TEST2 is entered in the COMMAND field, the second program TWIND2 is executed.

Program TWIND2:

```
DEFINE DATA LOCAL
1 COMMAND (A10)
END-DEFINE
*
DEFINE WINDOW TEST2
  SIZE 5*30
  BASE 15/40
  TITLE 'ANOTHER WINDOW'
  CONTROL SCREEN
  FRAMED POSITION SYMBOL BOT LEFT
INPUT WINDOW='TEST2'
  WITH TEXT 'message line'
  COMMAND (AD=U) /
  'dataline 1' /
  'dataline 2' /
  'dataline 3' 'long data line'
IF COMMAND = 'TEST'
  FETCH 'TWIND'
ELSE
  REINPUT 'invalid command'
END-IF
END
```

A second window is opened. The other window is still visible, but it is inactive.

```

message line
> r                                     > + Program      TWIND      Lib SAG
Bot  ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
0030 END-DEFINE
0040 *                                +----Sample Window-----+
0050 DEFINE WINDOW TEST              ! invalid command          ! Inactive
0060   SIZE 5*25                     ! COMMAND TEST2          ! Window
0070   BASE 5/40                     ! dataline 1             ! 
0080   TITLE 'Sample Window'         +More:      + >-----+
0090   CONTROL WINDOW
0100   FRAMED POSITION SYMBOL BOT LEFT
0110 INPUT WINDOW='TEST'
0120   WITH TEXT 'message line'
0130   COMMAND (AD=I) /
0140   'dataline 1' /                +-----ANOTHER WINDOW-----+ Currently
0150   'dataline 2' /                ! COMMAND                  ! Active
0160   'dataline 3' 'long data line' ! dataline 1                ! Window
0170 IF COMMAND = 'TEST2'           ! dataline 2                ! 
0180   FETCH 'TWIND2'               +More: + >-----+
0190 ELSE
0200   REINPUT 'invalid command'
0210 END-IF
0220 END
      ....+....1....+....2....+....3....+....4....+....5....+.... S 22   L 3

```

Note that for the new window the message line is now displayed on the full physical screen (at the top) and not in the window. This was defined by the CONTROL SCREEN statement in the TWIND2 program.

For further details on the statements DEFINE WINDOW, INPUT WINDOW and SET WINDOW, see the Natural Statements documentation.

Standard/Dynamic Layout Maps

As described in the section Tutorial - Using the Map Editor, a *standard layout* can be defined in the map editor. This layout guarantees a uniform appearance for all maps that reference it throughout the application.

When a map that references a standard layout is initialized, the standard layout becomes a fixed part of the map. This means that if this standard layout is modified, all affected maps must be re-cataloged before the changes take effect.

Dynamic Layout Maps

In contrast to a standard layout, a *dynamic layout* does not become a fixed part of a map that references it, rather it is executed at runtime.

This means that if you define the layout map as "dynamic" on the Define Map Settings For Map screen in the map editor (see the example below), any modifications to the layout map are also carried out on all maps that reference it. The maps need not be re-cataloged.

08:46:18				Define Map Settings for MAP		2001-01-22	
Delimiters				Format		Context	
-----				-----		-----	
Cls	Att	CD	Del	Page Size 23	Device Check
T	D		BLANK	Line Size 79	WRITE Statement	_
T	I		?	Column Shift	... 0 (0/1)	INPUT Statement	X
A	D		—	Layout STAN1	Help	_____
A	I)	dynamic Y (Y/N)	as field default	N (Y/N)
A	N		a	Zero Print N (Y/N)		
M	D		&	Case Default	... UC (UC/LC)		
M	I		:	Manual Skip N (Y/N)	Automatic Rule Rank	1
O	D		+	Decimal Char	Profile Name SYSPROF
O	I		(Standard Keys	.. Y (Y/N)		
				Justification	.. L (L/R)	Filler Characters	
				Print Mode _	-----	
				Control Var	Optional, Partial _
						Required, Partial _
						Optional, Complete	... _
						Required, Complete	... _
Apply changes only to new fields?				N (Y/N)			
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---							
Help		Exit		Let			

For further details on layout maps, see the section Map Editor.

Multilingual User Interfaces

Using Natural, you can create multilingual applications for international use.

Maps, help routines, error messages, programs, subprograms and copycodes can be defined in up to 60 different languages (including languages with double-byte character sets).

Below is information on:

- Language Codes
- Defining the Language of a Natural Object
- Defining the User Language
- Referencing Multilingual Objects
- Programs
- Error Messages
- Edit Masks for Date and Time Fields

Language Codes

Each language has a *language code* (from 1 to 60). The table below is an extract from the full table of language codes.

Language Code	Language	Map Code in Object Names
1	English	1
2	German	2
3	French	3
4	Spanish	4
5	Italian	5
6	Dutch	6
7	Turkish	7
8	Danish	8
9	Norwegian	9
10	Albanian	A
11	Portuguese	B

The language code (left column) is the code that is contained in the system variable *LANGUAGE. This code is used by Natural internally. It is the code you use to define the user language (see below). The code you use to identify the language of a Natural object is the *map code* in the right-hand column of the table.

Example:

The language code for Portuguese is "11".

The code you use when cataloging a Portuguese Natural object is "B".

For the full table of language codes, see the system variable *LANGUAGE as described in the Natural Programming Reference documentation.

Defining the Language of a Natural Object

To define the language of a Natural object (map, help routine, program, subprogram or copycode), you add the corresponding map code to the object name. Apart from the map code, the name of the object must be identical for all languages.

In the example below, a map has been created in English and in German. To identify the languages of the maps, the map code that corresponds to the respective language has been included in the map name.

Example of Map Names for a Multilingual Application:

DEMO1 = English map (map code 1)

DEMO2 = German map (map code 2)

Defining Languages with Alphabetical Map Codes

Map codes are in the range 1-9, A-Z or a-y. The alphabetical map codes require special handling.

Normally, it is not possible to catalog an object with a lower-case letter in the name - all characters are automatically converted into capitals.

This is however necessary, if for example you wish to define an object for Kanji (Japanese) which has the language code 59 and the map code "x".

To catalog such an object, you first set the correct language code (here 59) using the terminal command `%L=nn`, where *nn* is the language code.

You then catalog the object using the ampersand (&) character instead of the actual map code in the object name. So to have a Japanese version of the map DEMO, you store the map under the name DEMO&.

If you now look at the list of Natural objects, you will see that the map is correctly listed as DEMOx.

Objects with language codes 1-9 and upper case A-Z can be cataloged directly without the use of the ampersand (&) notation.

In the example list below, you can see the three maps DEMO1, DEMO2 and DEMOx. To delete the map DEMOx, you use the same method as when creating it, that is, you set the correct language with the terminal command %L=59 and then confirm the deletion with the & notation (DEMO&).

08:41:14		***** NATURAL LIST COMMAND *****						2001-01-25	
User SAG		LIST * *						Library SAG	
Cmd	Name	Type	S/C	SM	Vers	Level	User-ID	Date	Time
---	-----	-----	---	--	----	-----	-----	-----	-----
___	COM3	Program	S/C	S	2.2	0002	SAG	92-01-21	14:34:39
___	CUR	Program	+----- DELETE -----+					92-01-22	09:37:02
___	CURS	Map	!				!	92-01-22	09:37:41
___	D	Program	!	Please confirm deletion			!	92-01-21	14:13:14
___	DARL	Program	!	with name DEMOx			!	91-06-03	12:08:30
___	DARL1	Local	!		DEMO&_____		!	91-06-03	12:03:52
___	DAV	Program	+-----+-----+					92-01-29	09:07:52
de	DEMOx	Map	S/C	S	2.2	0002	SAG	92-02-25	08:41:04
___	DEMO1	Map	S/C	S	2.2	0002	SAG	92-01-22	08:38:32
___	DEMO2	Map	S/C	S	2.2	0002	SAG	92-01-22	08:07:32
___	DOWNCOM	Program	S	S	2.2	0001	SAG	91-08-12	14:01:10
___	DOWNCOMR	Program	S	S	2.2	0001	SAG	91-08-12	14:01:32
___	DOWNCOM2	Program	S	S	2.2	0001	SAG	91-08-15	13:02:20
___	DOWNDIR	Program	S	S	2.2	0001	SAG	91-08-16	08:03:56
From _____ (New start value)								0	
Command ==>									
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---									
Help		Exit		--		-		+	
								Canc	

Defining the User Language

You define the language to be used per user - as defined in the system variable *LANGUAGE - with the profile parameter ULANG (which is described in the Natural Parameter Reference documentation) or with the terminal command %L=*nn* (where *nn* is the language code).

Referencing Multilingual Objects

To reference multilingual objects in a program, you use the ampersand (&) character in the name of the object.

The program below uses the maps DEMO1 and DEMO2. The ampersand (&) character at the end of the map name stands for the map code and indicates that the map with the current language as defined in the *LANGUAGE system variable is to be used.

Example:

```
DEFINE DATA LOCAL
1 PERSONNEL VIEW OF EMPLOYEES
  2 NAME (A20)
  2 PERSONNEL-ID (A8)
1 CAR VIEW OF VEHICLES
  2 REG-NUM (A15)
1 #CODE (N1)
END-DEFINE
*
INPUT USING MAP 'DEMO&' /* <--- INVOKE MAP WITH CURRENT LANGUAGE CODE
...
```

When this program is run, the English map (DEMO1) is displayed. This is because the current value of *LANGUAGE is "1" = English.

MAP DEMO1

SAMPLE MAP

Please select a function!

1.) Employee information

2.) Vehicle information

Enter code here: _

In the example below, the language code has been switched to "2" = German with the terminal command %L=2.

When the program is now run, the German map (DEMO2) is displayed.

BEISPIEL-MAP

Bitte wählen Sie eine Funktion!

1.) Mitarbeiterdaten

2.) Fahrzeugdaten

Code hier eingeben: _

Programs

For some applications it may be useful to define multilingual programs. For example, a standard invoicing program, might use different subprograms to handle various tax aspects, depending on the country where the invoice is to be written.

Multilingual programs are defined with the same technique as described above for maps.

Error Messages

Using the Natural utility SYSERR, you can translate Natural error messages into up to 60 languages, and also define your own error messages.

Which message language a user sees, depends on the *LANGUAGE system variable.

For further information on error messages, see the Natural SYSERR Utility documentation.

Edit Masks for Date and Time Fields

The language used for date and time fields defined with edit masks also depends on the system variable *LANGUAGE.

For details on edit masks, see the session parameter EM as described in the Natural Reference documentation.

Skill-Sensitive User Interfaces

Users with varying levels of skill may wish to have different maps (of varying detail) while using the same application.

If your application is not for international use by users speaking different languages, you can use the techniques for multilingual maps to define maps of varying detail.

For example, you could define language code 1 as corresponding to the skill of the beginner, and language code 2 as corresponding to the skill of the advanced user. This simple but effective technique is illustrated below.

The following map (PERS1) includes instructions for the end user on how to select a function from the menu. The information is very detailed. The name of the map contains the map code 1:

MAP PERS1

SAMPLE MAP

Please select a function

1.) Employee information _

2.) Vehicle information _

Enter code: _

To select a function, do one of the following:

- place the cursor on the input field next to desired function and press ENTER
- mark the input field next to desired function with an X and press ENTER
- enter the desired function code (1 or 2) in the 'Enter code' field and press ENTER

The same map, but without the detailed instructions is saved under the same name, but with map code 2.

MAP PERS2

SAMPLE MAP

Please select a function

1.) Employee information _

2.) Vehicle information _

Enter code: _

In the example above, the map with the detailed instructions is output, if the ULANG profile parameter has the value 1, the map without the instructions if the value is 2.

Further details on ULANG are described in Profile Parameters in the Natural Parameter Reference documentation.

Dialog Design

This section tells you how you can design user interfaces that make user interaction with the application simple and flexible:

- Field-Sensitive Processing
*CURS-FIELD and POS(*field-name*)
 - Simplifying Programming
System Function POS
 - Line-Sensitive Processing
System Variable *CURS-LINE
 - Column-Sensitive Processing
System Variable *CURS-COL
 - Processing Based on Function Keys
System Variable *PF-KEY
 - Processing Based on Function-Key Names
System Variable *PF-NAME
 - Processing Data Outside an Active Window
System Variable *COM
 - Copying Data from a Screen
Terminal Commands %CS and %CC
 - Statements REINPUT/REINPUT FULL
 - Object-Oriented Processing
Natural Command Processor
-

Field-Sensitive Processing

*CURS-FIELD and POS(*field-name*)

Using the system variable *CURS-FIELD together with the system function POS(*field-name*), you can define processing based on the field where the cursor is positioned at the time the user presses ENTER.

*CURS-FIELD contains the internal identification of the field where the cursor is currently positioned; it cannot be used by itself, but only in conjunction with POS(*field-name*).

You can use *CURS-FIELD and POS(*field-name*), for example, to enable a user to select a function simply by placing the cursor on a specific field and pressing ENTER.

The example below illustrates such an application:

Example:

```

DEFINE DATA LOCAL
1 #EMP (A1)
1 #CAR (A1)
1 #CODE (N1)
END-DEFINE
*
INPUT USING MAP 'CURS'
*
DECIDE FOR FIRST CONDITION
  WHEN *CURS-FIELD = POS(#EMP) OR #EMP = 'X' OR #CODE = 1
    FETCH 'LISTEMP'
  WHEN *CURS-FIELD = POS(#CAR) OR #CAR = 'X' OR #CODE = 2

```

```
    FETCH 'LISTCAR'  
    WHEN NONE  
        REINPUT 'PLEASE MAKE A VALID SELECTION'  
    END-DECIDE  
  
END
```



```

                                SAMPLE MAP

                                Please select a function

                                1.) Employee information  _
                                2.) Vehicle information  _ ← Cursor positioned
                                                                on field

                                Enter code:  _

To select a function, do one of the following:

- place the cursor on the input field next to desired function and press ENTER
- mark the input field next to desired function with an X and press ENTER
- enter the desired function code (1 or 2) in the 'Enter code' field and press
  ENTER

```

If the user places the cursor on the input field (#EMP) next to Employee information, and presses ENTER, the program LISTEMP displays a list of employee names:

```

Page      1                                2001-01-22  09:39:32

                                NAME
                                -----

ABELLAN
ACHIESON
ADAM
ADKINSON
ADKINSON
ADKINSON
ADKINSON
ADKINSON
ADKINSON
ADKINSON
ADKINSON
ADKINSON
AECKERLE
AFANASSIEV
AFANASSIEV
AHL
AKROYD

```

Simplifying Programming

System Function POS

The system function `POS(field-name)` contains the internal identification of the field whose name is specified with the system function.

`POS(field-name)` may be used to identify a specific field, regardless of its position in a map. This means that the sequence and number of fields in a map may be changed, but `POS(field-name)` will still uniquely identify the same field. With this, for example, you need only a single `REINPUT` statement to make the field to be `MARKed` dependent on the program logic.

Note:

The values of `*CURS-FIELD` and `POS(field-name)` serve for internal identification of the fields only. They cannot be used for arithmetical operations.

Example:

```
...  
  DECIDE ON FIRST VALUE OF ...  
    VALUE ...  
      COMPUTE #FIELDX = POS(FIELD1)  
    VALUE ...  
      COMPUTE #FIELDX = POS(FIELD2)  
    ...  
  END-DECIDE  
  ...  
  REINPUT ... MARK #FIELDX  
  ...
```

Full details on `*CURS-FIELD` and `POS(field-name)` are described in Natural System Functions in the Natural Programming Reference documentation.

Line-Sensitive Processing

System Variable *CURS-LINE

Using the system variable *CURS-LINE, you can make processing dependent on the line where the cursor is positioned at the time the user presses ENTER.

Using this variable, you can make user-friendly menus. With the appropriate programming, the user merely has to place the cursor on the line of the desired menu option and press ENTER to execute the option.

The cursor position is defined within the current active window, regardless of its physical placement on the screen.

Note:

The message line, function-key lines and statistics line/infoline are not counted as data lines on the screen.

The example below demonstrates line-sensitive processing using the *CURS-LINE system variable. When the user presses ENTER on the map, the program checks if the cursor is positioned on line 8 of the screen which contains the option "Employee information". If this is the case, the program that lists the names of employees LISTEMP is executed.

Example:

```

DEFINE DATA LOCAL
1 #EMP (A1)
1 #CAR (A1)
1 #CODE (N1)
END-DEFINE
*
INPUT USING MAP 'CURS'
*
DECIDE FOR FIRST CONDITION
  WHEN *CURS-LINE = 8
    FETCH 'LISTEMP'
  WHEN NONE
    REINPUT 'PLACE CURSOR ON LINE OF OPTION YOU WISH TO SELECT'
END-DECIDE
END

```

Company Information

Please select a function

[] 1.) Employee information

2.) Vehicle information

Place the cursor on the line of the option you wish to select and press ENTER

The user places the cursor indicated by `[]` on the line of the desired option and presses ENTER and the corresponding program is executed.

Column-Sensitive Processing

System Variable *CURS-COL

The system variable *CURS-COL can be used in a similar way to *CURS-LINE described above. With *CURS-COL you can make processing dependent on the column where the cursor is positioned on the screen.

Processing Based on Function Keys

System Variable *PF-KEY

Frequently you may wish to make processing dependent on the function key a user presses.

This is achieved with the statement SET KEY, the system variable *PF-KEY and a modification of the default map settings (Standard Keys = "Y").

The SET KEY statement assigns functions to function keys during program execution. The system variable *PF-KEY contains the identification of the last function key the user pressed.

The example below illustrates the use of SET KEY in combination with *PF-KEY.

Example:

```
...  
SET KEY PF1  
*  
INPUT USING MAP 'DEMO&'  
IF *PF-KEY = 'PF1'  
    WRITE 'Help is currently not active'  
END-IF  
...
```

The SET KEY statement activates PF1 as a function key.

The IF statement defines what action is to be taken when the user presses PF1. The system variable *PF-KEY is checked for its current content; if it contains PF1, the corresponding action is taken.

Further details regarding the statement SET KEY and the system variable *PF-KEY are described in the Natural Statements and the Natural Programming Reference documentation respectively.

Processing Based on Function-Key Names

System Variable *PF-NAME

When defining processing based on function keys, further comfort can be added by using the system variable *PF-NAME. With this variable you can make processing dependent on the name of a function, not on a specific key.

The variable *PF-NAME contains the name of the last function key the user pressed (that is, the name as assigned to the key with the NAMED clause of the SET KEY statement).

For example, if you wish to allow users to invoke help by pressing either PF3 or PF12, you assign the same name (in the example below: INFO) to both keys. When the user presses either one of the keys, the processing defined in the IF statement is performed.

Example:

```
...
SET KEY PF3  NAMED 'INFO'
      PF12 NAMED 'INFO'
INPUT USING MAP 'DEMO&'
IF *PF-NAME = 'INFO'
  WRITE 'Help is currently not active'
END-IF
...
```

The function names defined with NAMED appear in the function-key lines:

Enter-PF1---	PF2---	PF3---	PF4---	PF5---	PF6---	PF7---	PF8---	PF9---	PF10--	PF11--	PF12---
		INFO									INFO

Processing Data Outside an Active Window

Below is information on:

- System Variable *COM
- Example Usage of *COM
- Positioning the Cursor to *COM - %T* Terminal Command

System Variable *COM

As stated above, only **one** window is active at any one time. This normally means that input is only possible within that particular window.

Using the *COM system variable, which can be regarded as a communication area, it is possible to enter data outside the current window.

The prerequisite is that a map contains *COM as a modifiable field. This field is then available for the user to enter data when a window is currently on the screen. Further processing can then be made dependent on the content of *COM.

This allows you to implement user interfaces as already used, for example, by Con-nect, Software AG's office system, where a user can always enter data in the command line, even when a window with its own input fields is active.

Note that *COM is only cleared when the Natural session is ended.

Example Usage of *COM

In the example below, the program ADD performs a simple addition using the input data from a map. In this map, *COM has been defined as a modifiable field (at the bottom of the map) with the length specified in the AL field of the Extended Field Editing . The result of the calculation is displayed in a window. Although this window offers no possibility for input, the user can still use the *COM field in the map outside the window.

Program ADD:

```

DEFINE DATA LOCAL
1 #VALUE1 (N4)
1 #VALUE2 (N4)
1 #SUM3 (N8)
END-DEFINE
*
DEFINE WINDOW EMP
  SIZE 8*17
  BASE 10/2
  TITLE 'Total of Add'
  CONTROL SCREEN
  FRAMED POSITION SYMBOL BOT LEFT
*
INPUT USING MAP 'WINDOW'
*
COMPUTE #SUM3 = #VALUE1 + #VALUE2
*
SET WINDOW 'EMP'
INPUT (AD=0) / 'Value 1 +' /
              'Value 2 =' //
              ' ' #SUM3
*
```

```

IF *COM = 'M'
  FETCH 'MULTIPLY' #VALUE1 #VALUE2
END-IF
END

```

Map to Demonstrate Windows with *COM

CALCULATOR

Enter values you wish to calculate

Value 1: 12__
Value 2: 12__

```

+-Total of Add-+
!               !
! Value 1 +     !
! Value 2 =     !
!               !
!           24  !
!               !
+-----+

```

Next line is input field (*COM) for input outside the window:

In this example, by entering the value "M", the user initiates a multiplication function; the two values from the input map are multiplied and the result is displayed in a second window:

Map to Demonstrate Windows with *COM

CALCULATOR

Enter values you wish to calculate

Value 1: 12__
Value 2: 12__

```

+-Total of Add-+
!               !
! Value 1 +     !
! Value 2 =     !
!               !
!           24  !
!               !
+-----+

```

```

+-----+
!               !
! Value 1 x     !
! Value 2 =     !
!               !
!           144  !
!               !
+-----+

```

Next line is input field (*COM) for input outside the window:

M

Positioning the Cursor to *COM - %T* Terminal Command

Normally, when a window is active and the window contains no input fields (AD=M or AD=A), the cursor is placed in the top left corner of the window.

With the terminal command %T*, you can position the cursor to a *COM system variable outside the window when the active window contains no input fields.

By using %T* again, you can switch back to standard cursor placement.

Example:

```
...  
INPUT USING MAP 'WINDOW'  
*  
COMPUTE #SUM3 = #VALUE1 + #VALUE2  
*  
SET CONTROL 'T*'  
SET WINDOW 'EMP'  
INPUT (AD=0) / 'Value 1 +' /  
              'Value 2 =' //  
              ' ' #SUM3  
...
```

Copying Data from a Screen

Below is information on:

- Terminal Commands %CS and %CC
- Selecting a Line from Report Output for further Processing

Terminal Commands %CS and %CC

With these terminal commands, you can copy parts of a screen into the Natural stack (%CS) or into the system variable *COM (%CC). The protected data from a specific screen line are copied field by field.

The full options of these terminal commands are described in the Natural Programming Reference documentation.

Once copied to the stack or *COM, the data are available for further processing. Using these commands, you can make user-friendly interfaces as in the example below.

Selecting a Line from Report Output for further Processing

In the following example, the program COM1 lists all employee names from Abellan to Alestia.

Program COM1:

```
DEFINE DATA LOCAL
1 EMP VIEW OF EMPLOYEES
  2 NAME(A20)
  2 MIDDLE-NAME (A20)
  2 PERSONNEL-ID (A8)
END-DEFINE
*
READ EMP BY NAME STARTING FROM 'ABELLAN' THRU 'ALESTIA'
  DISPLAY NAME
END-READ
FETCH 'COM2'
END
```

Page	1	2001-01-22 08:21:22
NAME		

ABELLAN		
ACHIESON		
ADAM		
ADKINSON		
ADKINSON		
ADKINSON		
ADKINSON		
ADKINSON		
ADKINSON		
ADKINSON		
AECKERLE		
AFANASSIEV		
AFANASSIEV		
AHL		
AKROYD		
ALEMAN		
ALESTIA		
MORE		

Control is now passed to the program COM2.

Program COM2:

```

DEFINE DATA LOCAL
1 EMP VIEW OF EMPLOYEES
  2 NAME(A20)
  2 MIDDLE-NAME (A20)
  2 PERSONNEL-ID (A8)
1 SELECTNAME (A20)
END-DEFINE
*
SET KEY PF5 = '%CCC'
*
INPUT NO ERASE 'SELECT FIELD WITH CURSOR AND PRESS PF5'
*   MOVE *COM TO SELECTNAME
FIND EMP WITH NAME = SELECTNAME
  DISPLAY NAME PERSONNEL-ID
END-FIND
END

```

In this program, the terminal command %CCC is assigned to PF5. The terminal command copies all protected data from the line where the cursor is positioned to the system variable *COM. This information is then available for further processing. This further processing is defined in the program lines shown in **boldface**.

The user can now position the cursor on the name that interests him; when he/she now presses PF5, further employee information is supplied.

SELECT FIELD WITH CURSOR AND PRESS PF5		2001-01-22 08:20:22
NAME		

ABELLAN		
ACHIESON		
ADAM	←	Cursor positioned on name for which more information is required
ADKINSON		
ADKINSON		
ADKINSON		
ADKINSON		
ADKINSON		
ADKINSON		
ADKINSON		
ADKINSON		
AECKERLE		
AFANASSIEV		
AFANASSIEV		
AHL		
AKROYD		
ALEMAN		
ALESTIA		

In this case, the personnel ID of the selected employee is displayed:

Page	1	2001-01-22 08:20:30
NAME	PERSONNEL ID	
-----	-----	
ADAM	50005800	

Statements REINPUT/REINPUT FULL

If you wish to return to and re-execute an INPUT statement, you use the REINPUT statement. It is generally used to display a message indicating that the data input as a result of the previous INPUT statement were invalid.

If you specify the FULL option in a REINPUT statement, the corresponding INPUT statement will be re-executed fully:

- With an ordinary REINPUT statement (without FULL option), the contents of variables that were changed between the INPUT and REINPUT statement will not be displayed; that is, all variables on the screen will show then contents they had when the INPUT statement was originally executed.
- With a REINPUT FULL statement, all changes that have been made after the initial execution of the INPUT statement will be applied to the INPUT statement when it is re-executed; that is, all variables on the screen contain the values they had when the REINPUT statement was executed.
- If you wish to position the cursor to a specified field, you can use the MARK option, and to position to a particular position within a specified field, you use the MARK POSITION option.

The example below illustrates the use of REINPUT FULL with MARK POSITION.


Example:

```
DEFINE DATA LOCAL
1 #A (A10)
1 #B (N4)
1 #C (N4)
END-DEFINE
*
INPUT (AD=M) #A #B #C
IF #A = ' '
    COMPUTE #B = #B + #C
    RESET #C
    REINPUT FULL 'Enter a value' MARK POSITION 5 IN *#A
END-IF
END
```

The user enters 3 in field #B and 3 in field #C and presses ENTER.

#A	#B	3	#C	3
----	----	---	----	---

The program requires field #A to be non-blank. The REINPUT FULL statement with MARK POSITION 5 IN *#A returns the input screen; the now modified variable #B contains the value 6 (after the COMPUTE calculation has been performed). The cursor is positioned to the 5th position in field #A ready for new input.

Enter name of field					
#A	—	#B	6	#C	0
					
Cursor positioned to 5th position in field					
Enter a value					

This is the screen that would be returned by the same statement, without the FULL option. Note that the variables #B and #C have been reset to their status at the time of execution of the INPUT statement (each field contains the value 3).

#A	—	#B	3	#C	3
----	---	----	---	----	---

Object-Oriented Processing

Natural Command Processor

The Natural Command Processor is used to define and control navigation within an application.

The Natural Command Processor consists of two parts: a development part and a runtime part.

- The development part is the utility `SYSNCP`. With this utility, you define commands and the actions to be performed in response to the execution of these commands. From your definitions, `SYSNCP` generates decision tables which determine what happens when a user enters a command.
- The run-time part is the statement `PROCESS COMMAND`. This statement is used to invoke the Command Processor within a Natural program. In the statement you specify the name of the `SYSNCP` table to be used to handle the data input by a user at that point.

For further information regarding the Natural Command Processor, see the Natural `SYSNCP` Utility documentation and the statement `PROCESS COMMAND` as described in the Natural Statements documentation.

Editors - General Information

This section gives an overview of which Natural objects are edited with which Natural editor. In addition, it contains information on Natural object names, split-screen mode and the editor profile.

You invoke a Natural editor with the system command EDIT as described in the Natural Command Reference documentation.

Which editor is invoked depends on the type of object you wish to edit:

- Programs, subprograms, subroutines, help routines, copycode and text are created and edited in the program editor.
- Global data areas, local data areas and parameter data areas are created and edited in the data area editor.
- Maps and help maps are created and edited in the map editor.
- Predict descriptions are edited in the Predict description editor (see the Predict documentation).

An online help system is provided with each editor.

Tutorials which introduce you to the main features of the editors are provided under Tutorial - Getting Started with Natural and Tutorial - Using the Map Editor.

In addition to the Natural editors, the Software AG Editor is provided as an optional feature, which is exclusively used by several Natural subproducts and other Software AG products (for further information, see the relevant section in the Natural Installation Guide for Mainframes and the Software AG Editor documentation).

Note:

If you wish to use the Software AG Editor as an alternative to the Natural program editor, Natural ISPF must be installed.

This section covers the following topics:

- Object Names
 - Split-Screen Mode
 - Editor Profile
-

Object Names

The name of a Natural object can be 1 to 8 characters long. It can consist of the following characters:

Character	Explanation
A - Z	upper-case alphabetical characters
0 - 9	numeric characters
-	hyphen
_	underline
/	slash
\$	dollar sign
&	ampersand (only as language code character; see also the section Defining the Language of a Natural Object)
#	hash/number sign
+	plus sign (only allowed as first character)

The first character of the name must be one of the following:

- an upper-case alphabetical character
- #
- +

If the first character is a hash/number (#) sign or a plus (+) sign, the name must consist of at least one additional character.

Split-Screen Mode

You can use all three Natural editors in split-screen mode: you can use one half of the screen for editing an object and at the same time have another Natural object displayed in the other half. Split-screen mode can be used to display a view, a data area, a Predict program description or a Natural program in the lower half of the screen. In addition, you can include items shown in the display section of the screen into the editing section that is, into the object you are currently editing.

Example:

The following figure shows the program editor in split-screen mode with the source code of a program in the editing section (upper half) and a local data area in the display section (lower half):

```

>                                     > + Program      SAGDEMO  Lib SAGTEST

Top      ....+....1....+....2....+....3....+....4....+....5....+....6....+....7.
0010 DEFINE DATA LOCAL USING L-INVOIC
0020                LOCAL USING L-INV-LN
0030 END-DEFINE
0040 *
0050 READ INVOICE-VIEW BY INVOICE-NO FROM 1
0060 *
0070   FIND INVOICE-LINE-VIEW WITH INVOICE-NO = INVOICE-NO (0050)
0080   DELETE
0090   END-FINE
0100 *
      ....+....1....+....2....+....3....+....4....+....5....+.. S 16   L 1
Split All      Local      L-INVOIC  Library SAGTEST
0010 V 1 INVOICE-VIEW                INVOICE
0020   2 CUST-NO                      N    8
0030   2 INVOICE-NO                   N    8
0040   2 DATE                         A    8
0050   2 AMOUNT                      N 9.2
0000
0000
0000
0000

```

Split-Screen Commands

The following commands can be used to display and position an object in split-screen mode. All commands begin with an **S** or with **SPLIT** to indicate the working mode - Split Screen. The **SPLIT** command is a cursor-sensitive command as described in the section Program Editor.

Command	Function
S ++	Position to bottom of object.
S B	
S - -	Position to top of object.
S T	
S +	Position one page forwards.
S +P	
S -	Position one page backwards.
S -P	
S + <i>nnn</i>	Position <i>nnn</i> lines forwards (only valid for program editor).
S - <i>nnn</i>	Position <i>nnn</i> lines backwards (only valid for program editor).
S .	Terminate split-screen mode.
S END	
S DATA <i>name</i> [<i>library</i>]	Display data area (global, local, parameter).
S DESCRIPTION <i>pgm-name</i> [<i>library</i>]	Display program description (if available) from the Predict Data Dictionary (valid for program and data area editor only).
S FUNCTION <i>name</i> [<i>library</i>]	Display the subroutine <i>name</i> , where <i>name</i> is the name of the subroutine as used in the DEFINE SUBROUTINE statement (not the name of the object containing the subroutine). This command is only available in the program editor.
S PROGRAM <i>name</i> [<i>library</i>]	Display program, subprogram, subroutine, help routine, copycode, text, map, class.
S SCAN [<i>value</i>]	Scan for a <i>value</i> . Each line containing the value is marked with a greater than (>) sign. To further scan for the same value, enter S SC only.
S VIEW <i>name</i> [SHORT]	Display view (DDM, as defined in Predict or SYSDDM). If SHORT is specified, the DDM is listed in short form (that is, only the Adabas short names and corresponding Natural field names are displayed) without any field header or field edit mask information.

In the data area editor, with DATA, PROGRAM and VIEW, an asterisk (*) can be used for *name* to display a list of all available objects. If the asterisk (*) is preceded by one or more characters, only those objects whose names begin with these characters are displayed.

A *library* can be specified with the program editor only. Under Natural Security, a library cannot be specified.

Editor Profile

This section covers the following topics:

- General Information
- Additional Options
- Editor Defaults
- General Defaults
- Color Definitions
- Direct Commands
- User Exit USR0070P
- Exit Profile Maintenance

General Information

When working with the Natural program editor or data area editor, an editor profile can be defined per user.

The editor profile shows the functions assigned to the PF and PA keys, and various other settings to be in effect during the edit session.

The profile can be modified by the users to suit their personal editing requirements.

To display your current profile, enter the command PROFILE in the command line of your program or data area editor. If such a profile does not exist, the default profile SYSTEM is displayed which can be used to create a user's profile. The SYSTEM profile is read from the user exit USR0070P and can be modified there.

To display the profile of another user or the default profile SYSTEM, enter the command PROFILE *profile-name*, where *profile-name* corresponds to the respective user ID.

When you are in an edit session and enter the PROFILE command together with your own user ID as profile name, your profile is always invoked directly from the database; any modifications made during the current session, but not yet saved on the database, will not apply. Therefore, to invoke your current session profile, enter the PROFILE command only.

When you enter the PROFILE command, the following screen is displayed:

10:36:42	***** NATURAL EDITORS *****	2001-01-30
- Editor Profile -		
Profile Name .. SAG_____		
PF and PA Keys		
PF1 ... --_____	PF2 ... -H_____	PF3 ... -_____
PF4 ... ++_____	PF5 ... +H_____	PF6 ... +_____
PF7 ... SCAN_____	PF8 ... _____	PF9 ... _____
PF10 .. SC=_____	PF11 .. *CURSOR_____	PF12 .. CANCEL_____
PF13 .. _____	PF14 .. _____	PF15 .. _____
PF16 .. _____	PF17 .. _____	PF18 .. _____
PF19 .. _____	PF20 .. _____	PF21 .. _____
PF22 .. _____	PF23 .. _____	PF24 .. _____
PA1 ... _____	PA2 ... _____	PA3 ... _____
Automatic Functions		
Auto Renumber .. Y	Auto Save Numbers .. 10_	Source Save into .. EDITWORK
Additional Options .. N		
Command ==>		
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---		
Help	Exit AddOp Save Flip	Del Canc

Attention:

Profile modifications made during the current session are lost when you enter the system command LOGON.

Entry	Explanation
Profile Name	<p>The name of the editor profile. Your own editor profile is displayed. If such a profile does not exist, you can modify the default profile to suit your own requirements. To do so, overtype the profile name SYSTEM with your user ID and save the renamed profile on the database.</p> <p>If you overtype the name of your profile with any other valid profile name (that is, any other valid user ID) and press ENTER, the profile of the corresponding user is invoked. Only one profile can be established per user ID, and any modifications made to another user's profile are only valid for the current session; they cannot be saved on the database.</p> <p>You can, however, overtype the profile name of another user's profile with your own user ID and then save the renamed profile on the database.</p>
PF and PA Keys	The commands assigned to the PF and PA keys are displayed. Any Natural editor or system command can be assigned. Combinations of commands (separated by a comma) are also possible.
Auto Renumber	<p>Y indicates that the source code in the program editor is to be renumbered automatically if any of the following occurs:</p> <ul style="list-style-type: none"> • a CATALOG, CHECK, RUN, SAVE or STOW command is issued; • a .I line command is issued and no line number is available for the line to be inserted. <p>Note: See also Renumbering of Source-Code Line Number References.</p>
Auto Save Numbers	<p>If a numeric value is entered, a copy of the source is saved automatically into the member specified in the "Source Save into" field after the specified number of modifications have taken place. Modification means each time that the source has been changed as a result of information entered on the screen.</p> <p>Auto Save Numbers applies to the map editor, too.</p>
Source Save into	The name of the member into which a copy of the source is to be saved automatically; the default name EDITWORK can be modified. The specified member is overwritten each time the number of changes specified in the "Auto Save Numbers" field has been exceeded.

Additional Options

If you mark Additional Options on the Editor Profile screen with **Y** or press PF4, a window will be displayed from which you can select the following options:

- Editor Defaults
- General Defaults
- Color Definitions

A plus (+) sign in front of an option indicates that some values have already been set in the corresponding window or via an appropriate editor command.

To select an option, you mark it with a **Y**.

For each option selected, a corresponding window will be displayed. The individual items of each window are explained below.

Editor Defaults

Option	Explanation
Escape Character for Line Command	The escape character which must precede each line command; the default escape character is a period (.).
Empty Line Suppression	<p>Y Any lines left blank are eliminated from the source as soon as you press ENTER.</p> <p>N Any lines left blank are not eliminated from the source when you press ENTER.</p> <p>This parameter only applies to the line command .I (see the sections Program Editor and Data Area Editor).</p>
Source Size Information	<p>Y The actual size of the object being edited and the remaining space available is displayed in the bottom information line of the editor screen. In addition, in the program editor, the programming mode (reporting or structured) is displayed in the top information line of the editor screen.</p> <p>N No such information is displayed.</p>
Source Status Message	<p>Y A transaction message will be displayed in the top information line each time the source is modified, checked, saved, cataloged or stowed.</p> <p>N No such transaction message will be displayed.</p> <p>The Source Status Message parameter only applies to the program editor.</p>
Absolute Mode for SCAN/CHANGE	<p>Y Corresponds to the editor command SET ABS ON.</p> <p>N Corresponds to the editor command SET ABS OFF.</p> <p>See Editor Commands in the sections Program Editor and Data Area Editor.</p>
Range Mode for SCAN/CHANGE	<p>Y Corresponds to the editor command SET RANGE ON.</p> <p>N Corresponds to the editor command SET RANGE OFF.</p> <p>See Editor Commands in the section Program Editor.</p>
Direction Indicator	Indicates the direction (+ or -) in which several editor commands are to work (see also Editor Command Line in the sections Program Editor and Data Area Editor).

General Defaults

Parameter	Explanation
Editing in Lower Case	<p>Y Lower-case characters in the source code are not automatically converted to upper case (corresponds to the terminal command %L).</p> <p>N Lower-case characters in the source code are automatically converted to upper case (corresponds to the terminal command %U). Automatic conversion is in effect by default.</p>
Dynamic Conversion of Lower Case	<p>This option is relevant only if the above option is set to Y:</p> <p>Y All lower-case characters in the source code are automatically converted to upper case - except text strings that are enclosed in apostrophes and comments: these remain as you enter them (see also the section Program Editor).</p> <p>N Any source code remains as you enter it.</p>
Position of Message Line	Indicates the position of the message line; possible values are TOP , BOT , <i>nn</i> and <i>-nn</i> .
Cursor Position in Command Line	Y Indicates that the cursor is positioned in the edit command line after the source has been modified and you pressed ENTER.
Stay on Current Screen	<p>Y Corresponds to the editor command SET STAY ON.</p> <p>N Corresponds to the editor command SET STAY OFF.</p> <p>See Editor Commands in the sections Program Editor and Data Area Editor.</p>
Prompt Window for Exit Function	Y When you enter the EXIT command in the editor command line, a confirmation window is displayed (see also Exit Function in the sections Program Editor and Data Area Editor).
ISPF Editor as Program Editor	Y Natural ISPF (if installed) is invoked instead of the Natural program editor.

Color Definitions

If you mark Color Definitions with **Y** in the Additional Options window, the following window will be displayed:

+-----COLOUR DEFINITIONS-----+		
!		!
! Edit Work Area	Split Screen Area	!
! Command Line NE		!
! Label Indicator NE	Label Indicator NE	!
! Line Numbers NE	Line Numbers NE	!
! Editor Lines NE	Editor Lines NE	!
! Scan and Error Line.. NE	Scan Line NE	!
! Information Text NE	Information Text ... NE	!
! Information Value ... NE	Information Value .. NE	!
! Information Line NE		!
+-----+		

In this window you can specify the colors in which the various parts of the edit-work and split-screen area of your program or data area editor are to be displayed.

To get a list of the colors available, you enter the question mark (?) help character in any of the input fields of the Color Definitions window or press PF1 (Help).

Apart from the Command and Information Lines and the corresponding Information Text and Values, the following individual parts can be assigned a specific color:

Label Indicator	Leftmost column of the editor screen; used, for example, to label a source code line on which a certain command has been performed (for example, the .X and .Y line commands).
Line Numbers	Column of the source code line numbers (program editor only).
Editor Lines	Lines of source code currently in the edit-work and/or split-screen area.
Scan and Error Line	All lines marked with an S (or a greater than (>) sign in split-screen mode) as a result of a scan operation, any line where an error was detected (marked with E and applicable in edit-work area of program editor only) and the error message line itself.

Direct Commands

The following direct commands can be used instead of the corresponding PF keys. Direct commands have to be entered in the command line at the bottom of the editor profile screen.

Command	Description
CANCEL	This command (or PF12) cancels the current function and returns you to the screen from which it was invoked. Any modifications made to the profile have no effect for the current session.
DELETE	This command (or PF11) deletes the current profile from the database. Before the profile is deleted, however, a confirmation window pops up, in which you can either type the name of the profile and press ENTER to confirm the deletion of the profile, or press ENTER only to exit the function.
EXIT	This command (or PF3) invokes the exit function prompt window, regardless of whether the corresponding editor default parameter (see General Defaults) is set or not.
FLIP	This command (or PF6 and PF18) is used to switch between the two PF-key lines.
REFRESH	This command (or PF13) displays the profile parameters currently valid for the session, which means that any modifications made so far, but not yet saved, are overwritten.
SAVE	This command (or PF5) saves all currently valid profile parameters both for the current session and on the database. However, it does not leave the current function.

User Exit USR0070P

The user exit routine USR0070P enables you to modify the parameter settings in the default profile SYSTEM. USR0070P provides a list of all parameters which are to receive a default setting.

With this user exit, you can also determine whether editor profiles are to be stored in the FNAT system file, the FUSER system file or the scratch-pad file.

Exit Profile Maintenance

To exit from any editor profile maintenance function, press PF3 (Exit) or enter the EXIT command in the command line at the bottom of your terminal screen. In both cases the EXIT Function prompt window is invoked offering you the following options:

Function	Explanation
Save and Exit	Returns you to the screen from where the current profile maintenance function was invoked and saves any modifications made to the current profile. Modifications are saved both for the current session and on the database. If you are working with another user's editor profile, however, modifications made to that profile cannot be saved on the database. They are valid for the current session only; a corresponding message is returned.
Exit without Saving	Returns you to the screen from where the current profile maintenance function was invoked. Any modifications made to the current profile are only valid for the current session; they are not saved on the database. Pressing ENTER corresponds to "Exit without Saving".
Resume Function	Closes the prompt window and returns you to the current profile maintenance function.

Program Editor

The Natural program editor is used to perform online full-screen editing of Natural source programs. With the program editor, you can create and edit source programs quickly and efficiently with a minimum of effort. This section describes how to use this editor:

- Invoking the Program Editor
- Top Information Line
- Bottom Information Line
- Editor Command Line
- Editing a Program
- Editor Commands
- Editor Commands for Positioning
- Line Commands
- Special PF-Key Functions
- Cursor-Sensitive Commands
- The Exit Function

Invoking the Program Editor

You invoke the program editor with the system command EDIT as described in the Natural Command Reference documentation.

When you invoke the program editor, the editor screen is displayed (as shown below with a program in the work area):

```

>                                     > + Program      SAGDEMO  Lib SAGTEST
All  ....+....1....+....2....+....3....+....4....+....5....+....Mode Structured.
0010 DEFINE DATA LOCAL USING L-INVOIC
0020             LOCAL USING L-INV-LN
0030 END-DEFINE
0040 *
0050 READ INVOICE-VIEW BY INVOICE-NO FROM 1
0060 *
0070   FIND INVOICE-LINE-VIEW WITH INVOICE-NO = INVOICE-NO (0050)
0080     DELETE
0090   END-FIND
0100 *
0110   DELETE
0120   END TRANSACTION
0130 END-READ
0140 *
0150 FETCH 'MENU'
0160 END
0170
0180
0190
0200
.....Current Source Size: 308 Char. Free: 44756   +.. S 16   L 1

```

Note:

If Natural ISPF is installed and the general editor profile default "ISPF Editor as Program Editor" is set to "Y", instead of the program editor, either the Natural ISPF main menu (if the EDIT command is entered without an object name) or the Natural ISPF editor screen with the specified object is invoked.

Top Information Line

The top information line of the editor screen is used to display a message indicating object modification. See also the section Source Status Message. In addition, the programming mode (structured or reporting) currently in effect is displayed. When a program is read into the edit area, the mode is set to the one which was in effect when the program was stowed. This information is only displayed if the "Source Size Information" parameter in the editor profile defaults is set to "Y".

Bottom Information Line

In the bottom information line of the editor screen, the following items of information are displayed:

Current Source Size	Size (number of characters) of the current object. As source lines are stored in variable length in the work area, trailing blanks within a source line are not counted; leading and embedded blanks are counted. This information is only displayed if the "Source Size Information" parameter in the editor profile defaults is set to "Y".
Char. Free	The number of characters still available in the work area. This information is only displayed if the "Source Size Information" parameter in the profile defaults is set to "Y".
S	Size (number of lines) of the object being edited.
L	The number of the source line currently displayed as the top line.

Editor Command Line

The top line of the program editor screen is the edit command line. In this line, you can enter:

- a Natural system command (for example, EDIT, CHECK, SAVE),
- one or more editor commands,
- the name of a Natural program to be executed.

Additionally, the following items of information are displayed:

Direction Indicator (+ or -)	The direction indicator can be set to control the direction of the editor commands ADD and SCAN and of the line commands ".C", ".I" and ".M". The value "+" indicates after and the value "-" indicates before . The exact interpretation is described with the relevant command description.
Object Type	The type of object currently in the work area. The object type can be changed by using the editor command SET TYPE.
Object Name	The name of the object currently in the work area.
Library (Lib)	The library to which you are currently logged on.

Editing a Program

Multiple Functions

Multiple functions can be performed from a single input screen:

- Source lines can be updated directly.
- One or more line commands can be used.
- One or more editor commands can be used.

The following restrictions related to multiple functions apply:

- Only one insert line command (.I) can be performed at a time.
- You can enter multiple commands in the command line of the editor: you can enter more than one editor command, but only the last command entered in the editor command line can be a Natural system command. For example:
SC 'MOVE',-2,RENUMBER.
- If you have changed the screen contents manually or by commands, a system command cannot be entered until you press ENTER.

Note:

Natural treats the editor command "N" like a system command.

Dynamic Conversion from Lower to Upper Case

When the Natural terminal command %L is set and dynamic conversion to upper case is specified, all source code you enter in the editor is automatically converted to upper case, with the following exceptions:

- Text strings that are not hexadecimal constants and are enclosed in apostrophes remain as you enter them.
- Text strings (with or without apostrophes) in objects of type Text remain as you enter them.
- Comments remain as you enter them.

Dynamic conversion from lower to upper case can be specified and deactivated in the editor profile.

Editor Commands

Editor commands are entered in the command line of the program editor. The command parameters must be separated either by the input delimiter character as defined with the Natural session parameter ID (the default delimiter character is comma ",") or by a blank. When multiple commands are entered, these must also be separated by the delimiter character or by blanks. Line commands must not be entered in the command line.

The following edit commands are available:

Editor Command	Function
<u>A</u> DD[(<i>n</i>)]	<p>This command adds <i>n</i> blank lines. If the direction indicator is set to "+", the lines are added after the last line of the object being edited; if the direction indicator is set to "-", the lines are added before the first line of the object.</p> <p>The value for <i>n</i> can be in the range from 1 to 9. If <i>n</i> is not (or not correctly) specified, 9 lines (4 in split-screen mode) are added by default.</p> <p>With the next ENTER, lines that are still left blank will be eliminated.</p>
CANCEL	With this command you leave the editor. Any modifications made since the last time the SAVE command was entered are not saved.
<u>C</u> HANGE	<p>This command scans for the value entered as <i>scan-value</i> and replaces each such value found with the value entered as <i>replace-value</i>. The syntax for this command is:</p> <p style="text-align: center;">CHANGE '<i>scan-value</i>' '<i>replace-value</i>'</p> <p>Any special character which is not valid within a Natural variable name can be used as the delimiter character.</p>
CLEAR	This command clears the edit area (including the line markers "X" and "Y").
DX, DY, DX-Y	This command deletes the X-marked line; or the Y-marked line; or the block of lines delimited by "X" and "Y". See also the line commands ".X" and ".Y".
EX, EY, EX-Y	This command deletes source lines from the top of the source area to, but not including, the X-marked line; or from the source line following the Y-marked line to the bottom of the source area; or all source lines in the source area excluding the block delimited by "X" and "Y". See also the line commands ".X" and ".Y".
EXIT	With this command you leave the editor.
LET	This command undoes all modifications made to the current screen since the last time ENTER was pressed. In addition, LET ignores all line commands already entered but not yet executed.
N [(<i>n</i>)]	<p>This command renumbers the source code lines of the program currently in the work area.</p> <p>If you only enter "N", the lines are numbered in increments of 10; if you enter "N (<i>n</i>)", the lines are renumbered in increments of <i>n</i>.</p> <p>If the value specified for "<i>n</i>" is too big, lines are numbered in increments of 5.</p> <p>Note: See also Renumbering of Source-Code Line Number References.</p>
PROFILE [<i>name</i>]	This command displays the current editor profile.
QUIT	Same as editor command CANCEL.

Editor Command	Function
REN ON OFF	<p>ON Renumbers a Natural source program whenever it is checked, run, saved, stowed or cataloged.</p> <p>OFF Indicates that automatic renumbering is not in effect.</p> <p>The default is ON (see also the section Editor Profile).</p> <p>Note: See also Renumbering of Source-Code Line Number References.</p>
<u>RESET</u>	This command deletes the current X and Y line markers and any marker previously set with the line command ".N". See also line commands ".X" and ".Y".
<u>SCAN</u> [<i>'scan-value'</i>]	<p>This command scans for data in the source area. If you enter SCAN without any parameter, the SCAN menu is invoked. If you enter SCAN '<i>scan-value</i>', a scan for <i>scan-value</i> is performed.</p> <p>If the supplied scan value is entered without delimiter characters, for example, "SCAN ABC D", the entire character string which follows the keyword SCAN is used as the scan value.</p> <p>SCAN is a cursor-sensitive command.</p>
<u>SCAN</u> = [+ -]	<p>This command scans for the next occurrence of the scan value. The direction of the scan operation is determined by the setting of the direction indicator.</p> <p>If the direction indicator is omitted or set to "+", the scan operation will be from the current position of the source area (top of the displayed source window) to the last line in the source area. If the direction indicator is set to "-", the scan operation will be backwards from the bottom line of the current screen to the first line in the source area. The direction for a given scan command can also be explicitly specified by entering "SCAN =+" or "SCAN =" prior to command execution.</p> <p>The first line which contains the scanned value is positioned to the top line after each SCAN command.</p> <p>Each line in which the scanned value is located is marked with an "S" to the left of the line.</p> <p>Note: The equal sign "=" used with the SCAN command is the default input assign character. If another character has been specified as input assign character (see session parameter IA as described in the Natural Parameter Reference documentation), that other character must be used instead.</p>
<u>SET ABS</u> [ON OFF]	<p>ON The SCAN and CHANGE commands operate in absolute mode, which means that the value to be scanned/changed need not be delimited by blanks or special characters.</p> <p>OFF The SCAN and CHANGE commands do not operate in absolute mode, which means that the value to be scanned/changed must be delimited by blanks or special characters.</p> <p>The default is OFF.</p>
<u>SET ESCAPE</u> <i>character</i>	The escape character which must precede each line command. The default escape character is ".".

Editor Command	Function
<u>SET NUL</u> [ON OFF]	<p>ON All occurrences of a value scanned with the SCAN command are deleted. After the deletion of the scanned value, the SET NUL command is automatically set to OFF.</p> <p>The default is OFF.</p>
<u>SET RANGE</u> [ON OFF]	<p>ON The SCAN and CHANGE commands operate in range mode, which means that the value to be scanned/changed must be located within the range of lines delimited by the X and Y line markers.</p> <p>OFF The SCAN and CHANGE commands operate in non-range mode, which means that no range limit is to be in effect.</p> <p>The default is OFF.</p>
SET SEQ [ON OFF]	<p>OFF If your input is numeric, the first four positions in the edit area are considered as the line number and are moved to the line number position once you press ENTER.</p> <p>This feature is useful, for example, if a statement line is to be referenced by a source code line number in another statement line; when you renumber the source code, the referencing line number is renumbered, too.</p> <p>ON Numeric input in the first four positions remains as entered.</p> <p>Except with object type Text, the default is OFF.</p>
<u>SET SIZE</u> [ON OFF]	<p>ON The program size is displayed at the bottom information line of the editor screen and the programming mode is displayed on the scale line.</p> <p>OFF This information is not displayed.</p> <p>The default is OFF.</p>
SET STAY [ON OFF]	<p>ON The current screen will stay when ENTER is pressed. Forward and backward positioning can be done by positioning commands only.</p> <p>OFF Pressing ENTER positions to the next screen.</p>
SET TYPE	<p>The object type is set automatically when an existing object is read into the work area.</p> <p>This command can be used to change the type of object to be edited:</p> <p>SET TYPE PROGRAM (Natural Program) SET TYPE SUBROUTINE (Natural Subroutine) SET TYPE SUBPROGRAM (Natural Subprogram) SET TYPE HELPROUTINE (Natural Help Routine) SET TYPE COPYCODE (Natural Copycode) SET TYPE TEXT (Natural Text) SET TYPE CLASS (Natural Class)</p>
<u>SHIFT</u> [- +nn]	<p>This command shifts each source line delimited by the X and Y markers to the left or right. The <i>nn</i> parameter represents the number of characters the source line is to be shifted. Comment lines are not shifted.</p>
<u>SHIFT</u> - -	<p>This command shifts each source line delimited by the X and Y markers to the leftmost position. Comment lines are not shifted.</p>

Editor Command	Function
<u>SHIFT</u> ++	This command shifts each source line delimited by the X and Y markers to the rightmost position (maximum 99 positions). Comment lines are not shifted.
STRUCT [DISPLAY]	This command performs structural indentation of a Natural source program. If DISPLAY is specified, the Natural source program is displayed in compressed form (see also the system command STRUCT in the Natural Command Reference documentation).
*	This command displays the editor command most recently entered.
*=	This command again executes the command most recently entered in the command line.
.	With this command you leave the editor. Any modifications made since the last time the SAVE command was entered are not saved.

See also Renumbering of Source-Code Line Number References in the Natural Reference Documentation.

Editor Commands for Positioning

Editor commands for positioning are entered in the command line of the program editor. The following commands are available for positioning:

Command	Function
+P	Position forwards one page.
+	
-P	Position backwards one page.
-	
+H	Position forwards half a page.
-H	Position backwards half a page.
T	Position to top of program.
--	
B	Position to bottom of program.
++	
+nnnn	Position forwards <i>nnnn</i> lines (maximum 4 digits).
-nnnn	Position backwards <i>nnnn</i> lines (maximum 4 digits).
nnnn	Position to line number <i>nnnn</i> .
X	Position to the line marked with "X".
Y	Position to the line marked with "Y".
POINT	Positions to the line in which the line command ".N" was entered. See also the line command ".P".

Line Commands

The line commands are listed below. The notation "(*nnnn*)" indicates a repetition factor. The default repetition value is 1 (with the exception of the ".I" command; see below).

Note:

You are recommended to enter a blank at the end of each line command. This prevents the editor from attempting to interpret any information existing on the line as part of the line command.

Line Command	Function
.C(<i>nnnn</i>)	Copies the line in which the command was entered.
.CX(<i>nnnn</i>) .CY(<i>nnnn</i>)	Copies the X-marked or the Y-marked line. See also the line commands ".X" and ".Y" as well as the notes in the following section.
.CX-Y(<i>nnnn</i>)	Copies the block of lines delimited by the X and Y markers. (See also the notes in the following section.)
.D(<i>nnnn</i>)	Deletes line or lines. The default is 1 line.
.I(<i>n</i>)	Inserts <i>n</i> empty lines, where <i>n</i> can be in the range from 1 to 9. If <i>n</i> is not (or not correctly) specified, 9 lines (4 lines in split-screen mode) are inserted by default. (See also the notes in the following section.)
.I(<i>obj,ssss,nnnn</i>)	Includes into the source an object contained in the current library or in the steplib (the default steplib is SYSTEM). Depending on the direction indicator, the object is inserted before or after the line in which you enter the command. If you wish to include only part of the object, you specify as <i>ssss</i> the first line to be included (e.g., "20" means the inclusion will start from the 20th line), and as <i>nnnn</i> the number of lines to be included. If you enter multiple commands, this command is always executed after all other line and/or editor commands have been executed. If the object is a map, an INPUT USING MAP statement with all defined variables is automatically included in the current line. If the object is a data area, the entire data area is included, except comment lines. Only stowed local and parameter data areas can be included into the source area; global data areas cannot be included.
.J	Joins the current line with the next line. If the resulting line exceeds the length of the editor screen line, the line is marked with "L" and must be split in two with the ".S" command (see below) before it can be modified.
.L	Undoes all modifications that have been made to the line since the last time ENTER was pressed.
.MX .MY	Moves the X-marked or the Y-marked line. See also the line commands ".X" and ".Y" as well as the notes below.
.MX-Y	Moves the block of lines delimited by the X and Y markers (see also the notes below).

Line Command	Function
.N	Marks (invisibly) a line to be positioned to the beginning of the source area by the editor command POINT. The mark is automatically deleted when an error with a line command or editor command occurs.
.P	Positions the line marked by this command to the top of the screen.
.S	Splits the line at the position marked by the cursor.
.X	Marks a line or the beginning of a block of lines, to be processed (see also the notes below).
.Y	Marks a line or the end of a block of lines, to be processed (see also the notes below).

Note:

If both the commands ".X" and ".Y" are applied to one line, it is treated as being marked with "X" and with "Y"; the line marker actually shown to reflect this status is a "Z".

If the direction indicator is set to "+", the copied, inserted or moved lines are placed after the line in which the corresponding command was entered; if the direction indicator is set to "-", the copied, inserted or moved lines are placed before the line in which the command was entered.

Special PF-Key Functions

The following special functions can also be controlled using PF keys:

Function	Explanation
*CURSOR	A line split function can be combined with the command ".I", ".CX", ".CX-Y", ".MX" or ".MX-Y". This is accomplished by assigning the value "*CURSOR" to a PF key. If this PF key is then pressed instead of ENTER after a line command has been entered, the line in which the command was entered is first split at the cursor position and then the line command is executed.
*X *Y	If a PF key is assigned the value "*X" or "*Y", the cursor position is marked X or Y whenever this PF key is used. These column markers are then used to determine which portion of a line is to be included in the command operation. See the example below.

Example:

```

                                XY
X      0010 MOVE A TO B
        0020 WRITE A B
Y      0030 MOVE B TO A
                                XY
        ....
        0100 .MX-Y.....

```

The block of text starting with the "A" in line 0010 and ending with the "B" in line 0030 is moved:

```

0010 MOVE
0030 TO A
....
0010 A TO B
0020 WRITE A B
0030 MOVE B

```

Cursor-Sensitive Commands

- The SCAN Commands
- The SPLIT Command
- The EDIT and LIST System Commands

Cursor-sensitive commands are commands where, instead of entering a name in the command line, you can mark the name with the cursor anywhere on the editor screen (except in the command line). You can place the cursor on any word that is not in the command line. It does not matter where on the word the cursor is placed.

The SCAN Commands

The SCAN [*scan-value*] command scans for data in the edit area. If the SCAN command is used without any parameter but with the cursor positioned outside the editor command line, this results in a scan operation for the string the cursor is positioned to. If the cursor is positioned to a blank character, however, the SCAN menu is invoked.

In split-screen mode, the cursor can be positioned to a string in the split-screen area, too. The scan operation, however, is performed in the edit area only.

When using the SPLIT SCAN [*scan-value*] command, the same applies as for the SCAN command, but the scan operation is performed in the split-screen area only (see also the section Split-Screen Commands).

Note:

To benefit from cursor sensitiveness as much as possible, the SCAN or SPLIT SCAN command should be assigned to a PF key.

The SPLIT Command

Instead of the commands SPLIT PROGRAM, SPLIT DATA, SPLIT FUNCTION and SPLIT VIEW, which you can use to display a programming object or DDM in the split-screen area of the editor (see also the section Split-Screen Commands), you only have to enter the command SPLIT and place the cursor on the name of the desired object. The object must be contained in the current library.

Note:

To benefit from cursor sensitiveness as much as possible, the SPLIT command should be assigned to a PF key.

The EDIT and LIST System Commands

The system commands EDIT and LIST are cursor-sensitive, too. Instead of specifying an object name, the cursor can be positioned to a text string of the object currently in the edit area that corresponds to the desired object name.

With the EDIT command, the corresponding object is loaded into the editor. If necessary, even a different editor is invoked.

With the LIST command, the corresponding object is listed, even if a view has been referenced.

For more information on EDIT and LIST see the Natural Command Reference documentation.

The Exit Function

If the editor default parameter "Prompt Window for Exit Function" is set to "Y", any time you enter the EXIT command in the command line, the EXIT Function prompt window is invoked, offering you the following options:

Option	Explanation
Save and Exit	Leaves the editor and saves all modifications made to the current object.
Exit without Saving	Leaves the editor without saving any modification made to the current object since the last SAVE command was entered.
Resume Function	Neither leaves the editor nor saves any modifications; the prompt window is closed and the current function is resumed.

When "Prompt Window for Exit Function" is set to "N", the EXIT command leaves the editor and saves all modifications made to the current object; no prompt window is displayed.

Data Area Editor

The Natural data area editor is used to define and maintain definitions for global, local and parameter data areas.

A data area definition can consist of user-defined variables, database views and global data blocks (a collection of variables and/or views).

This section covers the following topics:

- Invoking the Data Area Editor
- Top Information Line
- Bottom Information Line
- Editor Command Line
- Editing a Data Area
- Editor Commands
- Line Commands
- The Exit Function
- Defining Globally Unique IDs in the Local and Global Data Area Editors

Invoking the Data Area Editor

You invoke the data area editor with the system command EDIT, specifying a data area type (GLOBAL, LOCAL or PARAMETER) or the name of a data area with the command (for details, see the system command EDIT as described in the Natural Command Reference documentation). If you specify the name of a data area, it is read into the edit area of the data area editor.

The data area editor screen appears with a local data area in the edit area:

Local	TEST1	Library	SAGTEST	DBID	10	FNR	32
Command							> +
I T L	Name			F	Leng	Index/Init/EM/Name/Comment	
All	-	-----		-	----	-----	
*	LDA for new application						
1	INCOME			A	20	(1:3,1:5) INIT ALL<'0'>	
1	PERSON						
2	SEX			A	6		
2	AGE			N	3		
1	NAME			A	24		
R 1	NAME					/* REDEF. BEGIN : NAME	
2	FIRST-NAME			A	10		
2	MIDDLE-INIT			A	2		
2	LAST-NAME			A	10		
C 1	DOLLAR			A	5	CONST<' \$US'>	
V 1	FINANCE-VIEW					FINANCE	
2	PERSONNEL-NUMBER			N	8.0		
P 2	MAJOR-CREDIT					(1:1) /* PERIODIC GROUP	
3	CREDIT-CARD			A	18	(EM=XXX.XXX.XXX.XXX.XXX.XXX)	
3	CREDIT-LIMIT			N	4.0		
3	CURRENT-BALANCE			N	4.0		
-----	Current Source Size: 625	Free: 61408	-----	S 12	L 1		

Top Information Line

The top information line of the editor screen is used to display the type and name of the data area currently in the editor, as well as the library, database ID and file number to which you are currently logged on.

Bottom Information Line

In the bottom information line of the editor screen, the following items of information are displayed:

Current Source Size	Size (number of characters) of the current object. As source lines are stored in variable length in the work area, trailing blanks within a source line are not counted; leading and embedded blanks are counted. This information is only displayed if the "Source Size Information" parameter in the editor profile defaults is set to "Y".
Free	The number of characters still available in the work area. This information is only displayed if the "Source Size Information" parameter in the profile defaults is set to "Y".
S	Size (number of lines) of the object being edited.
L	The number of the source line currently displayed as the top line.

Editor Command Line

The second line of the data editor screen is the edit command line. In this line, you can enter:

- a Natural system command (for example, EDIT, CHECK, SAVE),
- one or more editor commands,
- the name of a Natural data area to be executed.

In addition, the direction indicator can be set to control the direction of several editor and line commands. The value "+" indicates **after** and the value "-" indicates **before**. The exact interpretation is described with the relevant command description.

Editing a Data Area

The editor screen of the data area editor is divided into columns of fields with the following possible entries:

Field	Explanation
I	<p>Label Indicator. Information field supplied by the editor. This column is not modifiable by the user. Possible entries are:</p> <p>E indicates that a definition error has been detected.</p> <p>I indicates that an initial value has been defined via the ".E" line command.</p> <p>M indicates that an edit mask has been defined via the ".E" line command.</p> <p>S indicates that both an initial value and an edit mask have been defined via the ".E" line command.</p> <p>Parameter Data Areas only (see also Edit Fields):</p> <p>blank indicates the parameter specification BY REFERENCE (default).</p> <p>V indicates the parameter specification BY VALUE.</p> <p>R indicates the parameter specification BY VALUE RESULT.</p>
T	<p>Type. Possible types are:</p> <p>B Data block</p> <p>C Constant (user-defined variable only) or Counter field (database field only)</p> <p>* Comment</p> <p>G Group</p> <p>M Multiple-value field</p> <p>O Handle of object</p> <p>P Periodic group</p> <p>R Redefinition</p> <p>U Globally Unique Identifier (GUID)</p> <p>V View</p>
L	<p>Level number (1 - 9). Variables which are not within a hierarchical structure must be assigned level 1. View definitions must be assigned level 1. Level numbers cannot be used with data block definitions.</p>
NAME	<p>Name of the variable, block or view.</p> <p>Instead of specifying a variable name, the filler option (<i>n</i>X) can be used. With the filler option, "<i>n</i>" filler bytes can be denoted within a field or variable being redefined, where "<i>n</i>" can be in the range from 1 to 253. The definition of trailing filler bytes is optional.</p>
F	<p>Format. Any format supported by Natural can be used.</p>
LENG	<p>Length. No length is permitted for formats C, D, T and L.</p>

Field	Explanation
INDEX/INIT/EM/ NAME/COMMENT	<p>This field can be used to define an array, to supply initial values for a variable or to supply an edit mask for a variable; for a view definition, the name of a DDM from which this view is derived must be entered; for a block definition, the name of the parent block must be entered; and a comment can be entered. See also the examples below.</p> <p>Together with an edit mask, also a field header (HD) and the print mode (PM) can be defined:</p> <p>(HD='Name' EM=XXX.XXX.XX PM=N)</p> <p>See the Natural Parameter Reference documentation for further information on the PM session parameter.</p> <p>Since this field may be too short to make all necessary or desired specifications, an additional Edit Fields facility is provided with the ".E" line command.</p> <p>Note: When defining a view, the name of the DDM from which this view is derived can be modified. However, this is only possible if all fields of the view are also contained in the DDM with the modified name.</p>

Examples of Array Definitions:

```
(2,2)    (2 dimensions, 2 occurrences)
(2,2,2)  (3 dimensions, 2 occurrences)
(1:10,2)
(-1:3,2)
```

Examples of Initial Value Assignments:

```
INIT<3>
INIT<'ABC'>
INIT<H'FF'>
CONST<12>
```

Example of an Edit Mask Definition:

```
(EM=999.99)
```

Editor Commands

The following editor commands can be entered in the command line of the data area editor:

Command	Function
<u>C</u> ATALOG [<i>name</i>]	This command catalogs the data area definition currently in the edit area.
<u>C</u> HECK	This command checks the data area definition currently located in the edit area. It also orders the entries INDEX/INIT/EM/NAME/COMMENT in the sequence shown on the editor screen.
CLEAR	This command clears the edit area.
<u>C</u> REATE <u>G</u> LOBALS	<p>This command can only be used for a library created using Natural Version 1.2. It collects all global variables contained in cataloged objects (not saved objects) of the current library and places them in a global data area named "COMMON".</p> <p>The asterisk notation can be used to restrict processing to only those objects whose names begin with the specified value. For example:</p> <pre>CREATE GLOBALS * (all objects) CREATE GLOBALS ABC* (all objects beginning with ABC)</pre>
EXIT	With this command you leave the data area editor.
<u>G</u> ENERATE [<i>name</i>]	This command generates Natural copycode using the data area definitions currently in the edit area. A DEFINE DATA LOCAL and corresponding END-DEFINE statement are automatically included. If a <i>name</i> is entered, the generated copycode is saved under this name.
PROFILE [<i>name</i>]	This command displays the current editor profile.
READ <i>name</i>	This command reads an existing data area definition into the edit area.
SET ABS [ON OFF]	<p>This command determines whether the SCAN command operates in absolute or non-absolute mode.</p> <p>ON: the SCAN command operates in absolute mode, which means that the value to be scanned need not be delimited by blanks or special characters.</p> <p>OFF: the SCAN command operates in non-absolute mode, which means that the value to be scanned must be delimited by blanks or special characters.</p> <p>The default is OFF.</p>
SET PREFIX <prefix>/off	<p>This command allows you to specify a prefix for field names.</p> <p>This prefix is then automatically placed before the value entered in the "Name" column for each line that is entered or modified, unless the name already begins with this prefix.</p> <p>If the concatenated variable is longer than 32 bytes, a message is given and the value in the name field can be shortened. If this is not done, the prefix will not be inserted.</p>

Command	Function
SET SCAN COMMENT NAME	<p>If SET SCAN is set to COMMENT, you can scan for a value in the "Comment" column.</p> <p>If SET SCAN is set to NAME, you can scan for a value in the "Name" column.</p> <p>You cannot scan in both columns simultaneously; the default is NAME.</p>
SET SIZE ON OFF	If SET SIZE is set to ON, the size of the data area is displayed at the bottom information line of the editor screen.
SET STAY ON OFF	<p>If STAY is set to ON, the current screen will stay when ENTER is pressed. Forward and backward positioning can be done by positioning commands only.</p> <p>If STAY is set to OFF, pressing ENTER positions to the next screen.</p>
SET TYPE	<p>This command sets the data area object type:</p> <p>G Global data area</p> <p>L Local data area</p> <p>P Parameter data area</p>
STOW [<i>name</i>]	This command saves and catalogs the data area definition currently in the edit area.

Line Commands

All line commands described for the Natural program editor (except those which require a line number) can be used in the data area editor as well.

You are recommended to enter a blank at the end of each line command. This prevents the editor from attempting to interpret any information existing on the line as part of the line command.

In addition, the following line commands are available for the data area editor:

Command	Function
.D	<p>This command deletes one or more lines.</p> <p>When entered for an individual field, only that field definition is deleted.</p> <p>When entered for a part of a hierarchical structure (view, group, redefinition), all subsequent definitions on subordinate levels are also deleted. If, for example, you enter ".D" for a group defined at level 2, everything belonging to that group and with a level number greater than 2 is also deleted up to (but not including) the next level 2 definition. Comment lines (which usually are not assigned a level) are also considered to be at a subordinate level. To avoid the undesired deletion of a comment, assign an appropriate level to it.</p> <p>Note: In the data area editor, the ".D" command works differently from the program editor.</p>
.D(<i>nnnn</i>)	This command deletes <i>nnnn</i> lines, beginning with the line in which you enter the command. Unlike ".D" (see above), ".D(<i>nnnn</i>)" affects only the number of lines specified, regardless of any hierarchical structure.
.E	<p>This command invokes a separate screen for the definition of initial values and edit masks.</p> <p>If ".E" is executed for a DDM field, the Edit Mask screen is invoked immediately, since only edit masks (and no initial values) can be defined.</p>
.F(<i>file-name</i>)	This command includes a Predict file (applicable to file types: Conceptual, Standard, Sequential, Other).
.I(<i>n</i>)	<p>This command adds <i>n</i> empty lines, where <i>n</i> can be in the range from 1 to 9. If <i>n</i> is not (or not correctly) specified, 10 lines (5 lines in split-screen mode) are added by default.</p> <p>If the direction indicator is set to "+", the lines are added after the current line of the object being edited; if the direction indicator is set to "-", the lines are inserted before the current line.</p>

Command	Function
<code>.I(obj)</code>	<p>This command includes a Natural object. Apart from data areas, the following object types can be specified:</p> <ul style="list-style-type: none"> programs, subprograms, subroutines, help routines, maps. <p>If the object specified as <i>obj</i> is not a data area, it must be available as cataloged object. A window appears in the data area editor screen where you can select one of the following data definitions to be incorporated into your current data area:</p> <ul style="list-style-type: none"> - all local variables and parameters contained in the specified object (including those incorporated from local and/or parameter data areas), - all local variables contained in the specified object (including those incorporated from local data areas), - only those local variables defined within the specified object, - all parameters contained in the specified object (including those incorporated from parameter data areas), - only those parameters defined within the specified object. <p>If you incorporate variable definitions from objects without a DEFINE DATA definition (that is, from objects coded in reporting mode), variable redefinitions (see the REDEFINE statement in the Natural Statements documentation) might be placed to a wrong position; that is, after the wrong variable. So, before compiling your new data area, check all variable definitions and redefinitions for correct positioning.</p> <p>If a variable redefinition results in more than one variable, each variable is incorporated as one individual redefinition using filler bytes where appropriate.</p> <p>If the specified object has been cataloged using the Natural Optimizer Compiler, initial values and constants cannot be incorporated.</p>
<code>.I(obj,ssss,nnnn)</code>	<p>This command includes a global, local or parameter data area. This feature is only supported for data areas which do not contain initial values or edit masks.</p> <p>The "ssss" entry can be used to indicate at which line the insertion is to begin. For example, when setting "ssss" to 20, the insertion begins with the 20th line of the data area. The "nnnn" entry can be used to indicate the number of lines to be inserted.</p> <p>If "ssss" and/or "nnnn" is specified for an object other than a data area (see the .I(obj) command), the specified value(s) are ignored.</p>
<code>.R</code>	<p>This command redefines a field or variable.</p> <p>With the filler option (<i>nX</i>), <i>n</i> filler bytes can be denoted within a field or variable being redefined. The definition of trailing filler bytes is optional.</p>

Command	Function
.V [(<i>view-name</i> [,NOFL])]	<p>This command defines a view.</p> <p>A view (DDM) layout is displayed. You then select the fields from the view which are to be used in the program.</p> <p>If no view name is specified, the view currently in the split screen is included.</p> <p>If ".V <i>view-name</i>" is specified within a view of the same name as specified for <i>view-name</i>, the selected fields are included in this view and no new view is defined.</p> <p>If NOFL is specified, the selected fields are included without format and length specification.</p> <p>When a periodic group or multiple-value field defined - in a DDM generated with Predict - as "PC" or "MC" respectively is included in a data area, a C* variable (internal count of occurrences) for the group or field is automatically generated and placed before the group or field. The index for such a periodic group or multiple-value field is defined with the number of occurrences defined in Predict. If the number of occurrences has not been defined in Predict, the maximum occurrences (191) are used.</p> <p>If Predict is active, Predict redefinitions and comments are incorporated, too.</p> <p>Note: With VSAM views, always the actual number of occurrences is displayed. In addition, VSAM views contain information on subdescriptors and superdescriptors (for further information, see the Natural for VSAM documentation).</p>
.*	<p>This command generates a C* variable for multiple-value fields or fields within a periodic group.</p>
<i>number</i> [(<i>nnn,m</i>)]	<p>This command is available in split-screen mode and with a view in the split-screen area only.</p> <p>To obtain fields and groups from the split-screen area, the level number of the field or group from the split-screen area must be specified in the first column (without a period "."). The field or group is inserted before or after the current line, depending on the setting of the direction indicator ("+" or "-"). Fields and groups from the split-screen area can be included as fields of a view (if <i>number</i> is entered inside a view) or as user variables.</p> <p>If the selected field has the same name as the field for which the command was entered, it is substituted instead of inserted.</p> <p>Multiple lines can be obtained from the split screen using the "<i>nnn</i>" notation where <i>nnn</i> is the number of lines to be included.</p> <p>The "<i>m</i>" notation can be used to specify a level number to be assigned to the field or group to be inserted.</p>

Note:

".I(obj.)", ".R" and ".*" are available in full-screen mode only, not in split-screen mode.

Edit Fields

To invoke the Initial Values and Edit Mask menu

- Enter the ".E" line command in front of a specific field.

This feature is not available for redefined fields.

17:11:57		***** EDIT FIELD *****		2000-07-12	
- Initial Values and Edit Mask -					
Local	SAGAREA	Library	SAGTEST	DBID	10 FNR 49
	Code	Function		Definition	
	-----	-----		-----	
	S	Single Value Initialization		no	
	F	Free Mode Initialization		no	
	E	Edit Mask Definition		no	
	P	Parameter Type		no	
	D	Delete all Definitions			
	?	Help			
	.	Exit			
	-----	-----		-----	
Code	?	for Field: FIELD1(A10/1:2)			

If any initial values or edit masks have been defined, the corresponding status message in the Definition column of the Initial Values and Edit Mask screen is changed from "no" to "yes".

The following functions are available:

Code	Function
S	<p>This function enables you to define an initial value for the specified field. You need only enter the desired field value; any further specifications necessary (including apostrophes for alphanumeric fields) are generated automatically. For an array (multiple-value field), an initial value can (but does not necessarily have to) be defined for each occurrence.</p> <p>With arrays, asterisk notation (*) can be entered in the command line to repeat the value in the last line of the previous page until the end of the current page.</p>
F	<p>This function, too, enables you to define an initial value for the specified field. However, a free-mode editor is provided where you can enter your initial value definitions according to the common Natural syntax definitions. In this way, for example, the same initial value can be assigned to a whole range of field occurrences at a time. During editing, however, the specified values are not checked (unless you enter the CHECK command).</p>
E	<p>This function enables you to define an edit mask and/or header for the specified field according to the Natural rules for edit mask specification.</p> <p>If both an edit mask and a header are specified, together they must not exceed 57 characters in length. However, if only an edit mask is specified, it can be up to 63 characters long; if only a header has been specified, it can be up to 58 characters long.</p> <p>If ".E" has been executed for a DDM field, this function is invoked immediately, since only edit masks (and no initial values) can be defined for DDM fields.</p>
D	<p>This function enables you to delete, at a stroke, all definitions made via the "S", "F" and "E" functions.</p> <p>Any "yes" status messages are changed to "no".</p>
P	<p>This function only applies to Parameter Data Areas and enables you to specify a parameter BY REFERENCE (default), BY VALUE or BY VALUE RESULT.</p> <p>See also Parameter-Data-Definition in the DEFINE DATA section of the Natural Statements documentation.</p>

Any definitions made within the Initial Values and Edit Mask function are immediately incorporated into the data area currently in your data area editor.

Special Commands Available within the Edit Fields Function

The following commands can be entered in the command line of any edit field subfunction:

Command	Function
EDIT	This command returns you to your data area editor screen.
.	This command returns you to the previous screen to continue processing.
- -	This command returns you to the beginning of the initial value specification(s). It is only available for arrays in Single Value Initialization mode.
+	This command takes you one page forward. If the last page has been reached or if there is only one page available, you are returned to your data area editor screen.
*	This command copies the initial value of the last occurrence of the previous page to all empty fields of the current page. It is only available for arrays in Single Value Initialization mode.

The ".E" Line Command with Control Variables

When the ".E" line command is entered in front of a control variable, the Define Attributes screen is invoked, where attributes and colors can be specified as initial values for control variables.

For details on attributes and colors, see the session parameters AD and CD in the Natural Parameter Reference documentation.

The Exit Function

If the editor default parameter "Prompt Window for Exit Function" is set to "Y", any time you enter the EXIT command in the command line, the EXIT Function prompt window is invoked, offering you the following options:

Option	Explanation
Save and Exit	Leaves the editor and saves all modifications made to the current object.
Exit without Saving	Leaves the editor without saving any modification made to the current object since the last SAVE command was entered.
Resume Function	Neither leaves the editor nor saves any modifications; the prompt window is closed and the current function is resumed.

When the parameter "Prompt Window for Exit Function" is set to "N", the EXIT command leaves the editor and saves all modifications made to the current object; no prompt window is displayed.

Defining Globally Unique IDs in the Local and Global Data Area Editors

The definition of Globally Unique IDs (GUIDs) requires NaturalX and is possible under TSO and OS/390 Batch only.

To define a GUID, enter "U" in the T(ype) column and fill the L(evel) and the Name columns.

Format and length will be inserted automatically when you confirm your entry.

If it is possible to generate the GUID in the current Natural environment, it will be inserted as a protected const in the free mode editing section.

If the GUID cannot be generated, the information will be displayed in the comment field and the free mode init field that it has not been inserted. In this case, you can insert the GUID into the source of the data area by using the interface USR2022 in the library SYSEXT.

Map Editor

The Natural map editor is used to create maps (screen layouts).

A map can be stored in the Natural system file, from where it can be invoked by a Natural program using an INPUT USING MAP statement (for input maps) or a WRITE USING MAP statement (for output maps).

This section covers the following topics:

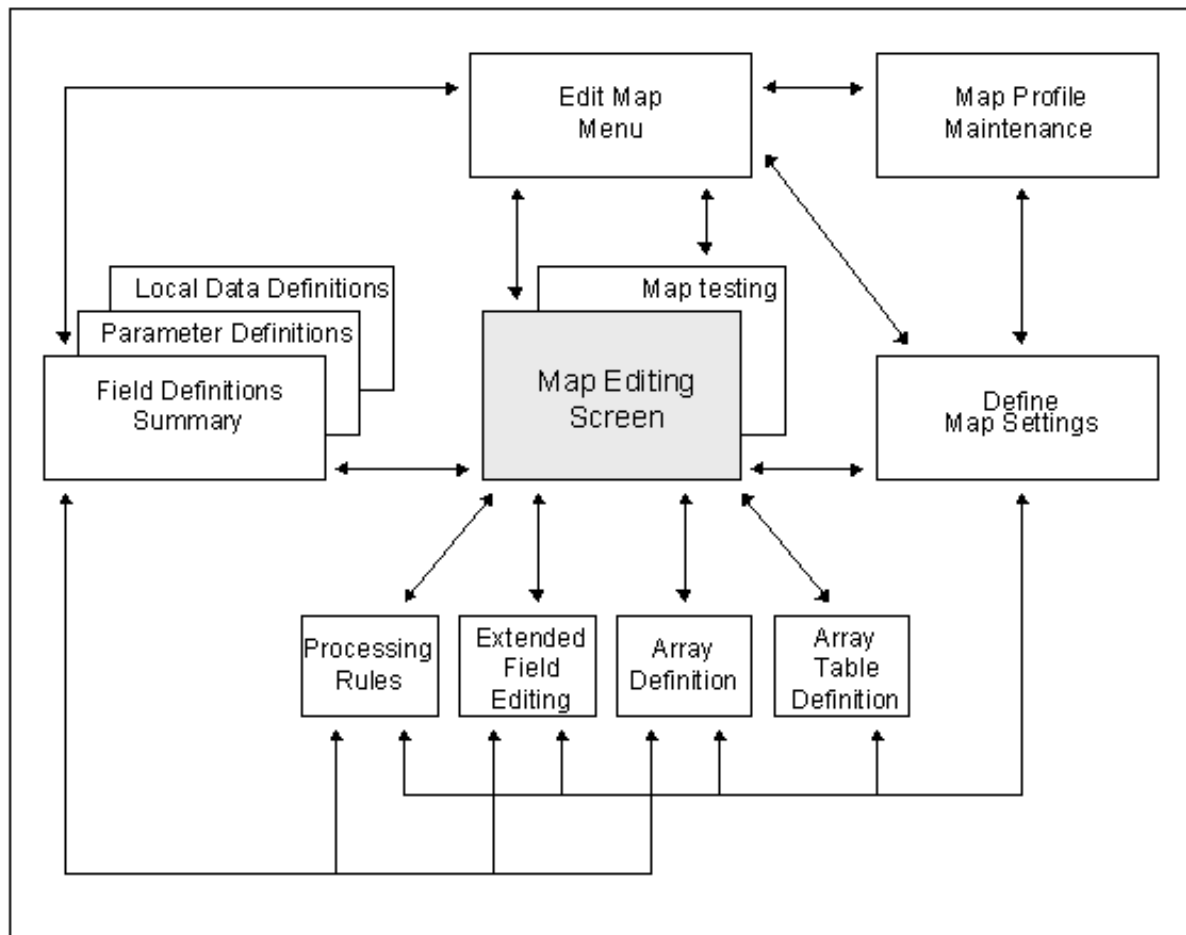
- Components of the Map Editor
- Summary of Map Creation
- Invoking the Map Editor
- Initializing a Map
- Editing a Map
- Defining Map Fields
- Extended Field Editing
- Post Assignment Function
- Array and Table Definition
- Processing Rules

See also:

- Tutorial - Using the Map Editor
-

Components of the Map Editor

The following figure provides an overview of the various components of the map editor and also shows the possible ways to get from one component to another:



Summary of Map Creation

Map creation involves four major steps:

Step 1

Definition of the map profile (that is, the field delimiters, format settings, context settings and filler characters to be used) by simply selecting the desired settings from a menu.

Step 2

Definition of the map. A map definition can be created before or after the data views that define the fields it contains. These two ways of creating a map definition are:

- First create a prototype map definition, next create the corresponding data views, then integrate the map into the application.
Fields can be defined directly on the screen. Each field is assigned a default name. Subsequently, when the corresponding data views have been created, the actual field definitions can be assigned to the map fields (post assignment).
- Create a map definition using existing data views.
If data views already exist, the map fields can be created by using the field definitions contained in the data views. In this case, all characteristics of a field defined in the data views are included when the field is positioned on the screen.

Step 3

Definition of the fields to be used in the map. A full set of map editing facilities is provided which permit simple and efficient map field definition:

- Full-screen or split-screen editing. In split-screen mode, the upper half of the screen is used for the display of user views or data definitions and the lower half for map definition. Map fields can be defined directly on the screen or can be selected from a user view or data definition.
- Screen positioning commands.
- Line commands, which are used to define tables and manipulate lines.
- Field commands, which are used to define arrays and manipulate fields.
- Editor facilities, which are used to edit processing (validation) rules.

Step 4

Storing the map definition. Once created, the map definition can be saved and/or cataloged in the Natural system file. Once saved, a map definition can be read and modified during a subsequent map editor session. Once cataloged, a map definition can be invoked from a Natural program.

Note:

The map editor uses the Auto Save Numbers function of the program and data area editors.

Invoking the Map Editor

You invoke the map editor with the system command:

EDIT MAP

If there is already a map in the source area, the map definition is displayed.

If the source area is empty, the Edit Map menu, which is the main menu of the map editor, is displayed:

```

16:49:52          ***** NATURAL MAP EDITOR *****          2001-01-17
User SAG              - Edit Map -                          Library SYSTEM

      Code      Function
      ----      -
      D      Field and Variable Definitions
      E      Edit Map
      I      Initialize new Map
      H      Initialize a new Help Map
      M      Maintenance of Profiles & Devices
      S      Save Map
      T      Test Map
      W      Stow Map
      ?      Help
      .      Exit

      Code .. I      Name .. _____      Profile .. SYSPROF_

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help      Exit Test Edit

```


The following entries appear on the Edit Map menu:

Entry	Explanation
User	The Natural user ID of the current user.
Library	The Natural library ID currently in effect.
Code	The code of the function to be executed (see below).
Name	<p>The source member which contains the map or help map.</p> <p>For multi-lingual maps, one digit of the source name should be reserved for the language code. For example:</p> <p style="padding-left: 40px;">USERMAP1 (language code is 1)</p> <p>The map above is called from the program by:</p> <p style="padding-left: 40px;">INPUT USING MAP 'USERMAP&'</p> <p>where "&" is replaced with the content of the system variable *LANGUAGE at execution time.</p>
Profile	<p>The session profile currently in effect.</p> <p>The profile name is set to the current library ID. If this profile ID is not available, it is set to the current user ID. If this profile ID is not available, the profile name is set to "SYSPROF".</p>

Overview of Functions

The following functions appear on the Edit Map menu:

Field and Variable Definitions

"Field Definitions" displays the following information for each map field:

- Field Name (name of the field)
- Field Mode (type of field), where:
 - D** means Data Area Field,
 - S** means System Variable,
 - U** means User-Defined Field,
 - V** means View Field,
 - blank* means Undefined Field
- Field Format (data type and field length)
- Field is an Array (A) or not ("blank")
- Number of attached Processing Rules
- Line and Column position.

This function is equivalent to line command "..E*" entered in the first map line.

The following commands are available within the Field Definitions subfunction:

Command	Description
A	Define array
D	Delete field
E	Edit map field
Prr	Edit processing rule
- -	Top
.	Exit

"Variable Definitions" displays all non-map field parameters and all local variables used in the map.

- Name
- Format
- The **Parameter Definitions** function is invoked by pressing PF9 on the Field Definitions screen; new parameters can be added and existing parameters can be modified.
- The **Local Data Definitions** function is invoked by pressing PF10 on the Field Definitions screen; new local variables can be added and existing variables can be modified. Local variables can be used to pass values from one processing rule to another.

The following commands are available within the two Variable Definitions subfunctions:

Command	Description
A	Define array
D	Delete variable
- -	Top
.	Exit

Note:

Command "D" does not delete a parameter if this parameter is still applied to any map field as a control variable, start value or help parameter.

Edit Map

Invokes the map editing screen to modify an existing map or help map definition.

The map editor starts an edit session in split-screen mode, where the upper half of the screen is used for user view definitions and the lower half for map definition. If the map being edited is a help map definition, full-screen mode is in effect.

Initialize a New Map

This function can be executed only if no object with the same name is stored in the Natural system file.

Initialize a New Help Map

This function should be used to create a help map, since it offers you the most flexibility when entering and editing text (leading blanks must be entered). It also provides additional checks to ensure that a valid help map is created.

The function can be executed only if no source and no object with the same name is present in the Natural system file.

A help map is stored as a map and can be referenced with the parameter "HE" in the map definition.

When initializing or editing a help map, you can specify in the map settings where the help map is to appear on the screen at execution time.

Maintenance of Profiles & Devices

This function allows you to add, modify or delete session, map and device profiles.

A session profile is used to assign default map settings to be used when a map or a help map is initialized.

A map profile defines the map settings to be in effect during map definition and execution.

A device profile defines the standard characteristics and settings for a device. This profile can be used to ensure compatibility between the map definition and the device to be used.

See also the section Context and setting Device Check.

Save Map

The map definition is stored in source form in the Natural system file.

Test Map

The current map definition is tested to ensure that it can be executed successfully. This includes testing of all processing rules and help facilities.

When testing a map, any additionally created numeric map parameters are initialized with the value 1.

Stow Map

Catalog (and save) a map definition. The map definition is cataloged and also stored in source form in the Natural system file.

Initializing a Map

This section describes the process of defining the map settings (profile) for a map or help map definition. When you select the function "Initialize New Map" or "Initialize a New Help Map", the first screen to be invoked is the Define Map Settings screen:

09:36:47				Define Map Settings for MAP				2001-01-17			
Delimiters				Format				Context			
-----				-----				-----			
Cls	Att	CD	Del	Page Size	23	Device Check	_____		
T	D		BLANK	Line Size	79	WRITE Statement				
T	I		?	Column Shift	...	1 (0/1)	INPUT Statement	X			
A	D		_	Layout	_____	Help	_____			
A	I)	dynamic	N (Y/N)	as field default	N (Y/N)			
O	D		+	Zero Print	N (Y/N)					
O	I		(Case Default	...	UC (UC/LC)					
M	D		&	Manual Skip	N (Y/N)	Automatic Rule Rank	1			
M	I		:	Decimal Char	Profile Name	SYSPROF		
				Standard Keys	..	N (Y/N)	Filler Characters				
				Justification	..	L (L/R)	-----				
				Print Mode	__	Optional, Partial	_		
				Control Var	_____	Required, Partial	_		
							Optional, Complete	...	_		
							Required, Complete	...	_		
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---											
Help				Exit				Let			

The Define Map Settings screen comprises the sections:

- Delimiters
- Format
- Context
- Filler Characters

Delimiters

Delimiters are used as a prefix to a field or a text constant to indicate class, attribute and color to be assigned to the field or text constant.

Valid classes are:

Class	Description
A	Input field
M	Output field which is modifiable
O	Output field which is not modifiable
T	Text constant

Valid attributes are:

Attribute	Description
B	Blinking
C	Cursive/italic
D	Default (non-intensified, non-blinking, etc.)
I	Intensified
N	Non-display
U	Underlined
V	Reversed video
Y	Dynamic (attributes to be assigned dynamically by a program)

Valid colors are:

Abbreviation	Color
BL	Blue
GR	Green
NE	Neutral
PI	Pink
RE	Red
TU	Turquoise
YE	Yellow

Any special character can be defined as a delimiter character - except the control character for terminal commands, the control character for map commands and the decimal notation character.

The *default* delimiter characters and their corresponding class and attribute settings are shown in the following table:

Class (Cls)	Attribute (Attr)	Delimiter (Del)
T (text constant)	D (default)	blank
T (text constant)	I (intensified)	?
A (input only field)	D (default)	_
A (input only field)	I (intensified))
O (output only field)	D (default)	+
O (output only field)	I (intensified)	(
M (modifiable field)	D (default)	&
M (modifiable field)	I (intensified)	:

These defaults can be changed by the Natural administrator by creating a session profile SYSPROF. They can be changed by the user by either creating own session profiles or changing map settings during the initialization of the map. This is done by simply entering the desired delimiter value in place of the default assignment.

See the section Defining Map Fields for examples of delimiter usage.

Format

The following map format settings can be used:

Entry	Explanation
Page Size	The number of map lines to be edited (1 - 250); if Standard Keys is set to "Y", the number of lines is restricted to 3 - 250. For a map which is output with a WRITE statement, you specify the number of lines of the logical page output with the WRITE statement, not the map size. Thus, the map can be output several times on one page.
Line Size	The number of map columns to be edited (5 - 249).
Column Shift	Column shift (0 or 1) to be applied to the map. This feature can be used to address all 80 columns on a 80-column screen (Column Shift = 1, Line Size = 80). Positional commands (PF10, PF11) must be used to edit all map positions.
Layout	The name of a map source definition which contains a predefined layout.
dynamic	Y Specifies the layout to be dynamic. The dynamically used layout does not become a fixed part of the map at compilation time, but is executed at runtime. Thus, subsequent modifications of a layout map become effective for all maps using that layout map. If the layout map includes user-defined variables, you have to define these parameters in the map using the layout map. Input fields and modifiable fields in the layout map are not open at runtime. Parameters can be added by pressing PF9 within the Field and Variable Definitions function. N Specifies the layout to be static. The static layout is copied into the source area when a map is initialized. Filler characters are not transferred; "N" is the default setting.
Zero Print	Y Displays a field value of all zeros as one zero only. N Displays a zero value as blanks; "N" is the default setting. This value is copied into the field definition when a new field is created and can be modified for individual fields using the extended field editing function.

Entry	Explanation
Case Default	<p>UC Indicates that all input entered for fields at map execution time is to be converted to upper case, that is, the session parameter AD=T is used as a field default.</p> <p>LC Indicates that no lower to upper case conversion is to be performed, that is, the session parameter AD=W is used as a field default. To make the value LC effective, you have to specify the value ON for the Natural profile parameter LC.</p> <p>This value is copied into the field definition when a new field is created and can be modified for individual fields using the extended field editing function.</p>
Manual Skip	<p>Y Does not automatically move the cursor to the next field in the map at execution time even if the current field is completely filled.</p> <p>N Moves the cursor automatically to the next field in the map at execution time when the current field is completely filled; "N" is the default setting.</p>
Decimal Char	The character to be used as the decimal notation character. This character can only be changed with the GLOBALS command.
Standard Keys	<p>Y Leaves the last two lines of the map empty so that function-key specifications can be entered at execution time.</p> <p>N Causes all lines to be used for the map; "N" is the default setting.</p>
Justification	<p>The type of field justification to be used for numeric and alphanumeric fields taken from a user view or data definition:</p> <p>L left justified</p> <p>R right justified</p> <p>This value is copied into the field definition when a new field is created.</p>
Print Mode	<p>The default print mode for variables:</p> <p>C Indicates that an alternative character set is to be used (special character table as defined by the Natural administrator).</p> <p>D Indicates that double byte character mode is to be used.</p> <p>I Indicates inverse print direction.</p> <p>N Indicates standard print direction.</p> <p>This value is copied into the field definition when a new field is created.</p>
Control Var	<p>The name of a control variable, the content of which determines the attribute characteristics of fields and texts that have the attribute definition AD=Y or (Y). The control variable referenced in the map must be defined in the program using that map.</p> <p>Removing a control variable from the format map settings implies that the control variable is removed from the map, too, unless it is associated to any other map field.</p>

Context

The following map context settings can be used:

Entry	Explanation
Device Check	If a device name is entered in this field, the map settings are checked for compatibility with the device profile of the specified device. If a setting is not compatible, a warning message is issued (see also the section Maintenance of Profiles & Devices).
WRITE Statement	Marking this field with a non-blank value produces a WRITE statement at the end of the map definition process. The resulting map can then be invoked from a Natural program using a WRITE USING MAP statement. Empty lines at the end of the map are automatically deleted so that the map can be output several times on one page.
INPUT Statement	Marking this field with a non-blank value causes the result of the map definition process to be an INPUT statement. The resulting map can then be invoked from a Natural program using an INPUT USING MAP statement.
Help	The name of a helproutine which is invoked at execution time when the help function is invoked for this map (global help for map). For a detailed explanation of the syntax, refer to the specification of the parameter "HE".
as field default	<p>Y Specifies that the helproutine for the map is to apply as default to each individual field on the map, which means that the name of each field is passed individually to the helproutine.</p> <p>N Specifies that the name of the map is passed to the helproutine; "N" is the default setting.</p> <p>Note: If you define the map settings for a help map, on the Define Map Settings for HELPMAP screen, the "Help" and "as field defaults" fields are replaced by the "Position Line Col" field described below.</p>
Position Line Col	<p>The position where the help map is to appear on the screen at execution time.</p> <p>This field only appears if you define the map settings for a help map. It replaces the "Help" and "as field defaults" fields on the Define Map Settings for HELPMAP screen.</p>
Automatic Rule Rank	The rank (priority) assigned to automatic Predict rules when they are linked to the map during field definition. Default is 1.
Profile Name	<p>The name of the profile which was active at map initialization time.</p> <p>If "ENFORCED" is displayed, the following map settings are protected:</p> <ul style="list-style-type: none"> ● all map delimiters ● static and dynamic layout ● device check ● WRITE and INPUT statements ● all filler characters ● automatic rule rank ● positioning of help maps <p>The name of the profile active at the time the map is created is stored within the map. When the map is edited later and another profile is active, a warning is produced but editing is allowed.</p>

Filler Characters

Filler characters can be assigned to indicate whether information for a field is mandatory and whether the field must be completely filled:

Field Type	Explanation
Optional, Partial	Input not mandatory, field need not be completely filled.
Required, Partial	Input mandatory, field need not be completely filled (AD=E).
Optional, Complete	Input not mandatory; if filled, field must be completely filled (AD=G).
Required, Complete	Input mandatory, field must be completely filled (AD=EG).

Filler characters can also be defined for individual fields using the extended field editing function. For definition of field types, see also the session parameter AD as described in the Natural Parameter Reference documentation.

Editing a Map

The map editor begins an edit session always in split-screen mode, which means that the upper half of the screen is used for user-view definitions and the lower half for map definition:

```

Ob  _                               Ob D CLS ATR DEL          CLS ATR DEL
.                                     .      T  D      Blnk      T  I      ?
.                                     .      A  D      _        A  I      )
.                                     .      A  N      1        M  D      &
.                                     .      M  I      :        O  D      +
.                                     .      O  I      (
.                                     .
001  --010---+-----+-----+---030---+-----+-----+---050---+-----+-----+---070---+--

```

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---

Help Mset Exit Test Edit -- - + Full < > Let

PF9 can be used to switch between full-screen and split-screen mode.

If the rightmost of the view definition windows does not contain a view, the current delimiter settings are displayed in this window instead.

Entering a period (.) in the first position of the leftmost view window returns you to the Edit Map menu.

The following commands and functions are available for editing a map:

- Commands and Function Keys for Positioning
- Line Commands
- Field Commands

Commands and Function Keys for Positioning

The commands and PF keys listed below can be used for map positioning on the screen; you enter the commands at the beginning of a map line:

Key	Command	Function
PF1		Invoke map editor help facility.
PF2		Display/modify the current map settings.
PF3	.Q	Terminate map editing and return to Edit Map menu.
PF4		Test the map definition (without Predict rules).
PF5		Invoke extended field editing for field at which the cursor is currently positioned.
PF6	.- -	Move to top of map.
PF7	.-	Move upwards half a window page.
	.- <i>nnn</i>	Move upwards <i>nnn</i> lines.
PF8	.+	Move downwards half a window page.
	.+ <i>nnn</i>	Move downwards <i>nnn</i> lines.
	.++	Move to bottom of map.
PF9	./	Switch between split-screen and full-screen mode.
PF10	.<	Move to the left half a window page.
	.< <i>nnn</i>	Move to the left <i>nnn</i> columns.
	.<<	Move to the left border of the map.
PF11	.>	Move to the right half a window page.
	.> <i>nnn</i>	Move to the right <i>nnn</i> columns.
	.>>	Move to the right border of the map.
PF12		Ignore changes made on screen subsequent to last use of ENTER.
	.*	Move top left corner to cursor position.

Line Commands

Line commands must be entered in the form "*..*line-command**" where "*..*" represents two occurrences of the control character in effect for the map definition.

It is recommended that you enter a blank at the end of each line command. This prevents the editor from attempting to interpret any information existing on the line as part of the line command.

The following line commands are available:

Command	Function
.. <i>A</i>	Array table definition.
.. <i>An</i>	Array table definition with <i>n</i> occurrences. This command can be used to create a table with <i>n</i> occurrences vertically for all fields specified in the current line.
.. <i>C</i>	Center a single line (that is, the line in which the command was entered). Two " <i>..C</i> " commands entered on the same screen center the first line and adjust the rest of the selected lines.
.. <i>Cn</i>	Center line and move the <i>n-1</i> lines below it accordingly.
.. <i>C*</i>	Center line and move all lines below it accordingly.
.. <i>D</i>	Delete a single line (that is, the line in which the command was entered). Two " <i>..D</i> " commands entered on the same screen delete the block of lines delimited by these commands.
.. <i>Dn</i>	Delete line and the <i>n-1</i> lines below it.
.. <i>D*</i>	Delete line and all lines below it. If the delete operation affects array elements the array is deleted in total.
.. <i>E</i>	Invoke the extended field editing function for all fields contained in the line. Two " <i>..E</i> " commands entered on the same screen display all fields within the range of lines delimited by these commands for possible extended field editing.
.. <i>En</i>	Invoke extended field editing for the line and the <i>n-1</i> lines below it.
.. <i>E*</i>	Invoke extended field editing for the line and all lines below it. The " <i>..E</i> " commands display a screen with the name and format of the requested fields. The field names shown can be modified. The CMD column can be used to select the desired function: extended field editing, array definition and processing rule editing.
.. <i>Fc</i>	Fill the empty spaces of a line with the character <i>c</i> .
.. <i>I</i>	Insert a single line. The last empty line on the screen is deleted in order to allow for the line insertion.
.. <i>In</i>	Insert <i>n</i> lines below the line in which the command was entered.
.. <i>I*</i>	Insert as many lines as possible below the command line.

Command	Function
..J	Join the line in which the command was entered with the line below it. Two "..J" commands entered on the same screen joins the range of lines delimited by the commands.
..Jn	Join the line in which the command was entered with the $n-1$ lines below it.
..J*	Join the line with all lines below it. If a join operation results in a line being too long, the lower line is split at the rightmost possible position and the left part is then joined with the previous line. The right part of the split line is then shifted to the left to align it with the line in which the command was entered.
..M	Move the line in which the command was entered below the cursor line. If two "..M" commands are entered on the same screen, the block of lines delimited by the commands is moved below the line marked with the cursor.
..Mn	Move the line and the $n-1$ lines below it below the line marked with the cursor.
..M*	Move the line in which the command is entered and all lines below it to the line below the line marked with the cursor. This command is only practical if the line marked with the cursor is above the line in which the command is entered.
..P	Invoke PF-key processing rule editing. PF-key processing rules are special processing rules to define activities assigned to program sensitive function keys.
..Pn	Invoke PF-key processing rule editing for rank level n .
..Q	Terminate map editing and return to the Edit Map menu.
..R	Repeat once all text constants on the line in which the command was entered. The cursor position is used to indicate the target line. If two "..R" commands are entered on the same screen, the text constants within the block of lines delimited by the commands are repeated.
..Rn	Repeat all text constants on this and the $n-1$ following lines. If the cursor is located below the command line, the same text is repeated n times.
..S	Split line at cursor position. If two "..S" commands are entered on the same screen, the block of lines delimited by the commands are split.
..Sn	Split the line where the command is entered and the $n-1$ lines below it at the cursor position.

Field Commands

Field commands must be entered in the form "*field-command*" where "." represents the control character in effect for the map definition. Each command must begin in the first position of a map field or text constant.

A field command can be applied to a range of fields or constants. A range can be specified in any of the following ways:

- Two or more of the same field commands can be used on the same screen. The column range (horizontal range) and the line range (vertical range) are determined by the positions of the commands. (The section Tutorial - Using the Map Editor provides examples which illustrate this.)
- A repetition factor n can be used. It can be enclosed within parentheses. The command is applied to the designated field and also to the fields in the $n-1$ lines below it. A repetition factor of "*" causes repetition until the bottom of the map is reached.

It is recommended that you enter a blank at the end of each field command. This prevents the editor from attempting to interpret part of the field as part of the field command.

The following field commands are available:

Command	Function
.A	<p>Define an Array. This command can be applied to a single field only and not to a range of fields.</p> <p>The array definition is specified on the screen provided. The resulting array is positioned with its left upper corner at the position where this command was entered.</p> <p>An array can be redefined by applying the ".A" command to one of its elements.</p>
.A> n	<p>Supply a repetition factor n with the ".A" command for the purpose of defining a one dimensional array (no spacing, no offsets) without having to use a separate screen.</p>
.C	<p>Center a field or range of fields between adjoining fields.</p> <p>To center a single field, enter ".C" in the field to be centered.</p> <p>To center a range of fields, enter ".C" in the first and last field to be centered, or enter ".C" in the first field and position the cursor to the last field to be centered.</p> <p>In the event that an adjoining field or fields are not present, the column boundaries in effect for the map definition are used instead.</p>
.D	<p>Delete a field or range of fields.</p> <p>To delete a single field, enter ".D" in the field to be deleted.</p> <p>To delete a range of fields, enter ".D" in the first and last field to be deleted. The field range to be deleted may extend beyond a single line. If an array element is deleted, the entire array is deleted.</p>
.E	<p>Invoke extended field editing for a field. This command can be applied only to a single field and not to a range of fields.</p> <p>Extended field editing can also be invoked by positioning the cursor to the selected field and pressing PF5.</p>
.J	<p>Join fields located on consecutive lines.</p> <p>The left boundary of the join operation corresponds to where the ".J" command is entered and the right one corresponds to the cursor position.</p>

Command	Function
.M	<p>Move a field or range of fields.</p> <p>To move a single field, enter ".M" in the field to be moved and place the cursor at the target position.</p> <p>To move a range of fields, enter ".M" in the first and last field to be moved and place the cursor at the target position.</p>
.P[<i>n</i>]	<p>Edit processing rules for a field.</p> <p>Supply a parameter <i>n</i> with the ".P" command to indicate the priority (rank) of the processing rule to be edited. If necessary, the value specified for <i>n</i> can be included in parentheses "()".</p>
.R	<p>Repeat (copy) a field or range of fields.</p> <p>To copy a single field, enter ".R" in the field to be copied and place the cursor at the target position.</p> <p>To copy a range of fields, enter ".R" in the first and last field to be copied and place the cursor at the target position.</p> <p>Repetition is always done downwards and from left to right. Fields generated by this command are assigned a dummy name. A valid name for each such field must be defined by using the post assignment function or the extended field editing function.</p> <p>Note: Arrays cannot be copied.</p>
.S	<p>Split (move) a line or a line range.</p> <p>Enter ".S" in the field at which splitting is to begin and place the cursor at the target position. The line is divided at the position where the ".S" command was entered. The right portion is then moved to the cursor position.</p>
.T	<p>Truncate (delete) a field or range of fields from a line.</p> <p>Enter ".T" in the field at which truncation is to begin. If this function is used to truncate (delete) an array element, the entire array is deleted.</p>

Defining Map Fields

The fields which are to comprise a map definition can be specified in any of the following ways:

- Defining Fields Directly on the Screen
- Selecting Fields from a User View or Data Definition
- Using System Variables in a Map Definition

Defining Fields Directly on the Screen

The fields which are to comprise the map definition are specified by entering a delimiter character followed by the number of positions to be allocated for the field. The following characters can be used:

Character	Meaning
9	Numeric position
0	Numeric right justified
.	Decimal notation (numeric field only)
S	Sign position (numeric field only)
HH	Hexadecimal (binary) (must be entered in groups of two)
X	Alphanumeric position

A repetition factor can also be specified in the form (*n*), for example, "X(5)" is equivalent to "XXXXX".

The following are examples of field definitions (the delimiter character can be changed as desired).

:999	3 positions, numeric
:000	3 positions, numeric right justified
:99.9	3 positions numeric with decimal point
:S9(6)	6 positions, signed numeric
:HHHH	4 positions, hexadecimal
:X	1 position, alphanumeric
:X(7)	7 positions, alphanumeric

Fields entered as shown above are assigned a dummy field name by the map editor. Each field must be assigned a name prior to map execution by using either the extended field editing or post assignment function. Other field formats can be specified using extended field editing.

Selecting Fields from a User View or Data Definition

A field can be selected from a user view or a data definition. The user view or data definition must first be specified next to the entry "Ob:" (object) on the screen (a second user view can also be specified on the right side of the screen).

To select a user view or data definition, first specify the object class and then the object name. Valid object classes are:

Class	Description
A	Parameter Data Area
C	Predict Conceptual Files (only if Predict is installed)
G	Global Data Area
H	Helproutine
L	Local Data Area
M	Map
N	Subprogram
P	Program
S	Subroutine
V	View

Programs, subroutines, subprograms and helproutines can only be used if they contain a DEFINE DATA statement. Data areas should only be used if they are STOWed.

Once a user view has been selected, it can be positioned forwards or backwards on the screen using positioning commands (+,-,++,--, +n,-n).

To include a user view field in the map definition, enter a delimiter character followed by the number (left-side view) or letter (right-side view) of the desired field. A group or items preceded by a period cannot be selected:

:3 (field 3 of the left-side view is selected)

:C (field C of the right-side view is selected)

Once all user view fields have been selected, press ENTER to show the fields on the map definition. If a selected field contains an edit mask, this is denoted by the notation "M".

The user view field name is used as the map field name for fields selected from a user view.

Using System Variables in a Map Definition

Natural system variables can also be specified in a map definition. The Natural Programming Reference documentation contains a complete description of Natural system variables.

A system variable must be preceded by an output delimiter:

(*TIME
(*DATE
(*APPLIC-ID

Extended Field Editing

Extended field editing is used to define field attributes.

Extended field editing is invoked by entering the line command `"..E"` or the field command `".E"`; the following screen is displayed:

```

Fld  START-NAME                                     Fmt  A8
-----
AD= MIT'_'_____ ZP=          SG=          HE= _____ Rls  0
AL= _____ CD=  ___ CV= _____ Mod User
PM=  ___ DF=          DY= _____
EM= _____

001  --010---+-----+-----+---030---+-----+-----+---050---+-----+-----+---070---+-----

      Please enter starting name .XXXXXXXXX

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  <--- --->  --  -      +      <      >      Let

```

It is possible to invoke extended field editing for the next or previous field in the map by pressing PF4 or PF5 respectively, or to invoke extended field editing for any field in the map by moving the cursor onto the desired field and pressing ENTER.

The sample screen above contains the following entries:

Entry	Explanation
Fld	<p>The field or array name.</p> <p>Field name assignment is related to the method with which the field was originally defined.</p> <p>If the field was taken from a user view or data definition, it is assigned the same name as the field in the user view or data definition.</p> <p>If the field was specified as a Natural system variable, it is assigned the name of the specified variable.</p>
Arr	<p>If the field is neither of the above, it is assigned a dummy name. You must assign a name to such a field prior to map execution. The name of a field can be changed. However, a prefix cannot be used for a field which did not have a prefix assigned previously. To obtain a prefixed field name, select the field from a user view or data definition.</p> <p>Note: Duplicate field names are only allowed for fields defined as "output only fields".</p> <p>See the section Defining Map Fields for additional information.</p>
Fmt	<p>The format and length of the field.</p> <p>These can be changed by overwriting the current entry.</p>
AL/FL/NL	The length to be used when displaying the field.
Rls	The number of processing rules currently defined for the field.
ZP	<p>Zero printing.</p> <p>OFF indicates that zero values for the field are not to be printed.</p> <p>ON indicates that zero values are to be printed.</p> <p>ZP appears on the screen only if the field is numeric.</p>
SG	<p>Sign position for numeric fields.</p> <p>OFF indicates that no sign position is to be allocated (default).</p> <p>ON indicates that a sign position is to be allocated.</p> <p>SG appears on the screen only if the field is numeric.</p>
PM	<p>Print Mode.</p> <p>C indicates that an alternative character set is to be used (as defined by the Natural administrator).</p> <p>D indicates that double-byte character set is to be used.</p> <p>I indicates inverse print direction.</p> <p>N indicates that it is not possible to print a hardcopy of the field content.</p>

Entry	Explanation
DF	<p>Date format (applies only to date fields):</p> <p>Determines the length of a date when converted to alphanumeric representation without an edit mask being specified:</p> <ul style="list-style-type: none"> S 8-byte representation with 2-digit year component and delimiters (<i>yy-mm-dd</i>). I 8-byte representation with 4-digit year component and no delimiters (<i>yyyymmdd</i>). L 10-byte representation with 4-digit year component and delimiters (<i>yyyy-mm-dd</i>). <p>For further information, see the session parameter DF as described in the Natural Parameter Reference documentation.</p>
DY	<p>Dynamic string attributes.</p> <p>The dynamic string parameter is used to define certain characters contained in the text string of an alphanumeric variable to control the attribute setting. See also the session parameter DY as described in the Natural Parameter Reference documentation.</p>
HE	<p>The name of a helproutine to be assigned to the field.</p> <p>For the syntax of the HE parameter, see below.</p> <p>For a detailed explanation of the operands used in the HE option, see the session parameter HE as described in the Natural Parameter Reference documentation.</p> <p><i>Operand1</i> can be the name of a helproutine specified in single quotes (') or a variable name.</p> <ul style="list-style-type: none"> ● If a field with the name specified as <i>operand1</i> in the HE option exists as a field of a map, the parameter references this field. ● If no field with that name exists in the map, the parameter is defined as A8 (default format assumed) in the map. <p>The format/length of <i>operand2</i> is defined in the following way:</p> <ul style="list-style-type: none"> ● If the parameter specified as <i>operand2</i> in the HE option is defined as a field of a map, the parameter references this field. ● If no field with that name exists, the parameter is defined as N7 (default format assumed) in the map. <p>Removing a parameter from the HE option implies that the parameter is also removed from the map, unless it is a map field or it is associated with any other map field as a help parameter or "Starting from" value.</p> <p>Non-map field parameters can be edited in the Field and Variable Definitions screen using PF9.</p> <p>Entering "HE=+" opens a window, which provides sufficient space for specifying multiple parameters to be passed to a helproutine.</p>
AD	<p>Field attributes.</p> <p>For source optimization reasons, the default values "D", "H", "F" and "W" are accepted but not retained (see also session parameter AD as described in the Natural Parameter Reference documentation).</p>
CD	<p>Color attributes.</p>

Entry	Explanation
CV	<p>Control variable for dynamic field attributes.</p> <p>The name of a variable which contains the attributes to be used for this field. This variable must be defined with format C in the program.</p> <p>The control variable also contains a MODIFIED data tag, which indicates whether the field has been modified following map execution.</p> <p>A single control variable can be applied to several map fields, in which case the MODIFIED data tag is set if any of the fields referencing the control variable has been modified.</p> <p>The control variable can be expanded up to three dimensions, for example, CONTR(*), CONTR(*,*), CONTR(*,*,*), depending on the rank of the corresponding array.</p> <p>Note: Removing a control variable from a field implies that the control variable is removed from the map, too, unless it is associated to any other map field.</p>
EM	Edit mask to be used for the field.
MODE	<p>Mode indicates how the field was created:</p> <p>DATA The field was created by selecting a field from a DEFINE DATA definition.</p> <p>SYS The field is a system variable.</p> <p>UNDEF The field was created directly on the screen and has a dummy name.</p> <p>USER The name of the field was created by extended field editing.</p> <p>VIEW The field was created by selecting a field from a view (file).</p>

HE Parameter Syntax:

$$HE = operand1 \left[, \left\{ \begin{matrix} operand2 \\ = \end{matrix} \right\} [, operand2] \dots 19 \right]$$

Operand	Possible Structure				Possible Formats										Referencing Permitted	Dynamic Definition
Operand1	C	S			A										no	no
Operand2	C	S			A										no	no

Post Assignment Function

A field which has been previously defined (in layout) directly on the screen can be assigned the field name and field attributes of a user view field or a DEFINE DATA definition.

Note:

Duplicate field names are only allowed for fields defined as "output-only fields".

A map field which has been created using a DDM field definition can be redefined using the field definition from a view defined in a data area.

Post assignment can be done by entering the user view field number (or letter) as shown in the view window directly behind the delimiter of the field.

This function can only be used if the formats of the layout agrees with the field definition. N and P are considered to be identical numeric.

This function cannot be used for view arrays if one or more dimensions of that array are smaller than the dimensions of the array in the layout.

If a length conflict occurs, an AL/FL/NL attribute is generated to map the field definition to the layout definition with truncation or expansion. Data are truncated when AL/FL/NL is specified.

Array and Table Definition

The array definition function is used to define the occurrences and layout of an array.

Array definition is initiated by the field command ".A" or by issuing the line command "..E" and then marking the desired field with the function code "A".

The table definition function is used to define the occurrences and layout of more than one array at the same time. The arrays must begin in the same map line.

Table definition is invoked by the line command "..A".

Below you will find information on:

- Array Definition
- Table Definition

Array Definition

The upper portion of the following screen is displayed for the purpose of array definition:

Name #001	Upper Bnds 1____ 1____ 1____		

Dimensions	Occurrences	Starting from	Spacing
0 . Index vertical	1____	_____	0 Lines
0 . Index horizontal	1____	_____	1 Columns
0 . Index (h/v) V	1____	_____	0 Cls/Ls
001 --010---+-----+-----030---+-----+-----050---+-----+-----070---+-----			
Please enter starting name .AXXXXXXXXXXXXXXXXXXX			
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---			
Help Mset Exit -- - + < > Let			

You can specify the following:

Entry	Explanation
Upper Bnds	<p>Indicates the upper bounds of the array; that is, the highest occurrence in (from left to right) the first, second and third dimension.</p> <p>If a field defined in a program is used to define the map array, the upper bounds of that field (user-defined variable or database field), as defined in the program, are used; these cannot be overwritten on the array definition screen.</p> <p>If the map array is derived from a user view array or a data definition, the dimensions of the map array must not exceed the dimensions shown in this field.</p> <p>If the map array is not derived from a user view array or a data definition, the dimensions of the map array must not exceed the dimensions as defined in the Natural program.</p>
Dimensions	An array can have up to three dimensions. The order in which the dimensions of the array are mapped to the map layout is determined by the values entered to the left of the Index operands.
Occurrences	The number of occurrences to be defined for a dimension.
Starting From	<p>The starting index value for a dimension. A numeric value can be used, or a variable name can be used to indicate that the actual value is supplied in the Natural program which invokes the map definition.</p> <p>If the variable is not defined otherwise as a field in the map, it is assumed to be of format/length N7. If so, it can be edited using PF9 in the Field and Parameter Definition screen.</p> <p>Note: Removing a "Starting from" value from an array implies that the variable is removed from the map, too, unless it is a map field or it is associated to any other map field as a "Starting from" value or help parameter. To edit "Starting from" values press PF9 in the Field and Variable Definitions screen.</p>
Spacing	The number of blank lines (for vertical dimensions) or blank columns (for horizontal dimensions) to be inserted between each dimension occurrence.

Examples of Array Definitions

Example 1:

A one-dimensional array consisting of 10 vertical occurrences with 2 blank lines to be inserted between each occurrence.

Name #001	Upper Bnds 10____ 1____ 1____		

Dimensions	Occurrences	Starting from	Spacing
1 . Index vertical	10_	_____	2 Lines
0 . Index horizontal	1__	_____	1 Columns
0 . Index (h/v) V	1__	_____	0 Cls/Ls

Example 2:

Same as example 1 except that the array is to be horizontal.

Name #001	Upper Bnds 10____ 1____ 1____		

Dimensions	Occurrences	Starting from	Spacing
0 . Index vertical	1__	_____	0 Lines
1 . Index horizontal	10_	_____	1 Columns
0 . Index (h/v) V	1__	_____	0 Cls/Ls

Example 3:

A two-dimensional array. The first dimension consists of 10 vertical occurrences with 1 blank line between each occurrence. The second dimension consists of 5 horizontal occurrences with 2 blank columns between each occurrence.

Name #001	Upper Bnds 10____ 5____ 1____		

Dimensions	Occurrences	Starting from	Spacing
1 . Index vertical	10_	_____	1 Lines
2 . Index horizontal	5_	_____	2 Columns
0 . Index (h/v) V	1_	_____	0 Cls/Ls

Example 4:

Same as example 3 except that the order of the dimensions is reversed.

Name #001	Upper Bnds 5_____ 10_____ 1_____		

Dimensions	Occurrences	Starting from	Spacing
2 . Index vertical	10_	_____	1 Lines
1 . Index horizontal	5_	_____	2 Columns
0 . Index (h/v) V	1_	_____	0 Cls/Ls

Example 5:

A three-dimensional array. The first dimension consists of 3 vertical occurrences with 1 blank line between each occurrence. The second dimension consists of 5 horizontal occurrences with 2 blank columns between each occurrence. The third dimension consists of 2 occurrences, expanded vertically within each occurrence of the first dimension.

Name #001	Upper Bnds 3_____ 5_____ 2_____		

Dimensions	Occurrences	Starting from	Spacing
1 . Index vertical	3__	_____	1 Lines
2 . Index horizontal	5__	_____	2 Columns
3 . Index (h/v) V	2__	_____	0 Cls/Ls

Example 6:

An example using "Starting from". The first dimension consists of 10 vertical occurrences starting from index I. 'I' is defined in the map editor with format/length N7 by default. The second dimension consists of 5 horizontal occurrences starting from the index 3.

Name #001	Upper Bnds 10__ 5__ 1__		

Dimensions	Occurrences	Starting from	Spacing
1 . Index vertical	10_	I_____	1 Lines
2 . Index horizontal	5__	3_____	2 Columns
0 . Index (h/v) V	1__	_____	0 Cls/Ls

Example 7:

An example of making a two-dimensional display from a one-dimensional array. The array consists of 40 elements. It is displayed in two columns with 20 lines each. This is achieved by specifying 0 as the horizontal index.

Name #001	Upper Bnds 40____ 1____ 1____		

Dimensions	Occurrences	Starting from	Spacing
1 . Index vertical	20_	_____	0 Lines
0 . Index horizontal	2__	_____	10 Columns
0 . Index (h/v) V	1__	_____	0 Cls/Ls

The sample screen above contains the following entries:

Entry	Explanation
Main Index	The number of vertical occurrences, the starting position and the number of lines to be skipped between each dimension occurrence.
Second Index	<p>The direction (horizontal or vertical), the starting position and the number of lines/columns to be skipped between each dimension occurrence.</p> <p>The second dimension only applies if one of the arrays has more than one dimension. In this case the second dimension can be displayed either horizontally (in which case there must be enough space in the line for all selected occurrences) or vertically (in which case there must be enough lines on the map to display main dimension times second dimension occurrences, including line spacing).</p>
Third Index	<p>The direction (horizontal or vertical), the starting position and the number of lines/columns to be skipped between each dimension occurrence.</p> <p>The third dimension only applies if one of the arrays has more than two dimensions. In this case the third dimension can be displayed either horizontally (in which case there must be enough space in the line for all selected occurrences) or vertically (in which case there must be enough lines on the map to display main dimension times second dimension times third dimension occurrences, including line spacing).</p>
Name of Variable	All names of field arrays contained in the table are displayed.
Col Pos	The column position in which the field is located. This is displayed for informational purposes only.
Dimension Size	The size of the array as defined in a user view or data definition, or as in a Natural program. If the map array is derived from a user view array or data definition, the dimensions of the map array must not exceed the dimensions shown in this field. If the map array is not derived from a user view array, the dimensions of the map array must not exceed the dimensions as defined in the Natural program.
Order	The order in which the dimensions are to be defined. M, S and T correspond to Main, Second and Third.
2nd Ind Occ.	The number of occurrences to be defined for the second index.
3rd Ind Occ.	The number of occurrences to be defined for the third index.

Example of Table Definition:

```

DEFINE DATA
  1 ARRAY1 (A3/1:10)
  1 ARRAY2 (A5/1:10,1:2)
  1 ARRAY3 (A7/1:10,1:2,1:3)
END-DEFINE

```

Table definition:

16:52:39	***** NATURAL MAP EDITOR *****										2001-01-17
- Array Table Definition -											
Main Index:	Vert. Occur.	2	Starting from _____				Spacing	2	Lines		
Second Index:	Direction(H/V)	V	_____					1	Cls/Ls		
Third Index:	Direction(H/V)	V	_____					0	Cls/Ls		

Name of Variable (truncated)	Col Pos	Dimension	Size	Order	2.Ind	3.Ind					
		Ind1	Ind2	Ind3	M S T	Occ.	Occ.				

ARRAY1	3	10	1	1	1						
ARRAY2	32	10	2	1	1 2	2					
ARRAY3	58	10	2	3	1 2 3	2	3				

ARRAY1 is a one-dimensional array with ten occurrences. The first two occurrences are expanded in the table.

ARRAY2 is a two-dimensional array. The first index consists of ten occurrences and the second index consists of two occurrences. The first two occurrences of the first index and both occurrences of the second index are expanded in the table.

ARRAY3 is a three-dimensional array. The first index consists of ten occurrences, the second index consists of two occurrences and the third index consists of three occurrences. The first two occurrences of the first index, both occurrences of the second index and all three occurrences of the third index are expanded in the table.

Table layout:

(*DATE	Map containing an array table of multi-dimensional arrays			(*TIME
ARRAY1 (1-dim.)	ARRAY2 (2-dim.)	ARRAY3 (3-dim.)		
:xxxxxxxxxx	:xxxxxxxxxx	:xxxxxxxxxx	Third Index	
	Second Index	:xxxxxxxxxx	(3 vertical	
Main Index	(2 vertical	:xxxxxxxxxx	occurrences)	
(2 vertical	:xxxxxxxxxx	:xxxxxxxxxx		
occurrences)		:xxxxxxxxxx		
		:xxxxxxxxxx		
:xxxxxxxxxx	:xxxxxxxxxx	:xxxxxxxxxx		
		:xxxxxxxxxx		
	:xxxxxxxxxx	:xxxxxxxxxx		
		:xxxxxxxxxx		
		:xxxxxxxxxx		
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--				
Help Mset Exit Test Edit Top - + Full < > Let				

The table is defined as a collection of arrays which share the following characteristics:

- The number of occurrences of the main index must be the same for each array of the table. The main index is always expanded vertically.
- All elements of a specific index must be placed in the same line. Thus, spacing between the elements of a specific index depends on the array with the largest dimension.

Processing Rules

- Field-Related Processing Rules
- Function-Key-Related Processing Rules
- Processing Rule Editing

Field-Related Processing Rules

Three types of processing rules can be defined:

- Inline processing rules
- Free Predict rules
- Automatic Predict rules

Inline processing rules are defined within a map source and do not have a name assigned. The availability of Predict is not required for inline rules.

Free Predict rules have a name assigned and are stored in Predict.

To edit a free processing rule, enter the rule during map creation and assign a name to it. Inline rules can become Predict rules (and vice versa) by assigning/removing the rule name.

Predict automatic rules apply to database fields and are defined by the Predict administrator. When a field is created by selecting it from a view or a data definition, and if the field is a database field, all automatic rules for that field are linked to the map definition. All automatic rules are concatenated and treated as a single map rule.

The rank of the automatic rules is defined in the map settings (default 1).

Automatic rules cannot be modified using the map editor. They can, however, be assigned a different rank either by using the command "P=*n*" or by just overwriting the old rank.

If Predict rules are modified subsequently by the Predict administrator, or new automatic rules are linked to a database field, or automatic rules are removed, it is sufficient to recatalog the map.

Note:

If a field with linked Predict processing rules is renamed, the rules are lost and must be linked again.

An ampersand "&" within the source code of a processing rule is dynamically substituted with the fully qualified name of the field using the rule; this does not apply if the rule is used by individual array elements.

Example:

```
IF & = ' ' REINPUT 'ENTER NAME' MARK *&
```

The field name notation "&*field-name*" within the source code of a processing rule allows you to have DDM-specific rules that cross-check the integrity of values between database fields, without having to explicitly qualify the fields with a view name. As *field-name* you specify the name of the database field as defined in the DDM, and at compilation time, Natural dynamically qualifies the field by replacing the "&" with the corresponding view name. This allows you to use the same processing rule for specific fields, regardless of which view the fields are taken from.

Function-Key-Related Processing Rules

Two types of function-key-related processing rules can be defined:

- Inline processing rules
- Free Predict rules

Function-key-related processing rules can be used to assign activities to program sensitive function keys during map processing. For function keys which already have a command assigned by the program, this command is executed without any rule processing.

Example:

```
IF *PF-KEY = 'PF3'  
    ESCAPE ROUTINE  
END-IF
```

When this rule is executed, map processing is terminated without further rule processing.

Processing Rule Editing

Processing-rule editing is invoked by the field command ".P", or by issuing the line command "..E" and then placing the function code "P" next to the field for which processing rule editing is to be performed. PF-key processing rule editing is invoked by the command "..P".

A parameter can be used (.Prr) to indicate the rank (priority) of the processing rule to be defined/edited. A field can have up to 100 processing rules (rank 0 to 99). At map execution time, the processing rules are executed in ascending order by rank and screen position of the field. PF-key processing rules are always assumed to have the first screen position.

For optimum performance, the following assignments are recommended when assigning ranks to processing rules:

Rank	Processing Rule
0	Termination rule
1 - 4	Automatic rules
5 - 24	Format checking
25 - 44	Value checking for individual fields
45 - 64	Value cross-checking between fields
65 - 84	Database access
85 - 99	Special purpose

How to Select a Rule for Editing

If you enter the field command ".P*" in a map field, you obtain a list of all processing rules defined for the field.

If you enter the line command "..P*" in any map line, you obtain a list of all function-key-related processing rules defined for the map.

On each list, the Predict rules are identified by their names, the inline rules by their first three source code lines. From each list you can select a rule for editing by entering its rank.

The screen for processing rule editing (with a processing rule example) is shown below:

```

Variables used in current map                                MOD
MODTXT(A3)                                                  U
FVAR(A75/1:6)                                              U
FTYP(A1/1:6)                                              U
RULEMODE(A6)                                              U
RULE-NAME(A32)                                            D
FIELDAN(A5)                                              D

Rule _____ Field FULCB3.CBCOM
>                > + Rank 0      S 1   L 1   Struct Mode
ALL  ....+....10...+....20...+....30...+....40...+....50...+....60...+....70..
0010 *
0020 IF & EQ MASK('?')
0030 REINPUT USING HELP
0040 END-IF
0050 *
0060
0070
0080
0090
0100
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help  Mset  Exit  Test      --    -    +    Full  Sc=      Let

```

During processing rule editing you can switch between split-screen and full-screen mode using PF9 or SPLIT/SPLIT E command. The upper half of the split screen displays the definitions of all map fields (except system variables). This display can be positioned by split-screen commands.

The source code used to define the processing rule is entered/edited in the same way as with the Natural program editor.

While working in the processing-rule editor, processing rules can be edited by entering the following commands in the editor command line:

Command	Function
<u>A</u> DD[(<i>n</i>)]	This command adds <i>n</i> empty lines in source code. See the more detailed description of the ADD command in the section Editor Commands.
<u>C</u> HANGE ' <i>string1</i> ' <i>string2</i> '	This command scans for the value entered as <i>string1</i> and replaces each such value found with the value entered as <i>string2</i> .
<u>C</u> HECK	This command checks the rule.
<u>C</u> LEAR	This command clears the edit area (including the line markers "X" and "Y").
DX, DY, DX-Y	This command deletes the X-marked line; or the Y-marked line; or the block of lines delimited by "X" and "Y".
EX, EY, EX-Y	This command deletes source lines from the top of the source area to, but not including, the X-marked line; or from the source line following the Y-marked line to the bottom of the source area; or all source lines in the source area excluding the block of lines delimited by "X" and "Y".
EXIT	This command terminates the rule editing function and return to map editing.
.	
P	This command positions forward to the next rule defined for the field.
P*	This command selects a rule from the selection menu.
Prr	This command selects the rule with rank <i>rr</i> .
P= <i>rr</i>	This command changes the rank of a processing rule to rank <i>rr</i> .
<u>P</u> OINT	This command positions the line in which the line command ".N" was entered to the top of the current screen.
<u>R</u> ESET	This command deletes the current X and/or Y line markers and any marker previously set with the line command ".N".
<u>S</u> AVE <i>name</i>	This command saves a rule as copycode with the name <i>name</i> .
<u>S</u> CAN [<i>'scan-value'</i>]	This command scans for data in the source area. Entering SCAN without any parameter displays the SCAN menu. Entering SCAN ' <i>scan-value</i> ' results in a scan for <i>scan-value</i> .
<u>S</u> CAN = [+ -]	This command scans for the next occurrence of the scan value. The direction of the scan operation is determined by the setting of the direction indicator. See the more detailed description of the SCAN commands in the section Editor Commands.
<u>S</u> HIFT [- + <i>nn</i>]	This command shifts each source line delimited by the X and Y markers to the left or right. " <i>nn</i> " represents the number of characters the source line is to be shifted. Comment lines are not shifted.
<u>S</u> HIFT - -	This command shifts each source line delimited by the X and Y markers to the leftmost position. Comment lines are not shifted.
<u>S</u> HIFT ++	This command shifts each source line delimited by the X and Y markers to the rightmost position (maximum 99 positions). Comment lines are not shifted.

Command	Function
<u>S</u> PLIT [<u>E</u> ND]	This command switches between split-screen mode and full-screen mode (see also the section Split-Screen Commands).
<u>T</u> EST	This command tests a map.
<u>U</u> NLINK	This command unlinks an inline or Predict free rule from the field.

Note:

To select a rule from all free Predict rules, enter a "?" in the rule name field of the processing rule editing screen.

Editor Commands for Positioning

Editor commands for positioning are also entered in the command line of the rule editor. The following commands are available:

Command	Function
+P	Position forwards one page.
+	
-P	Position backwards one page.
-	
+H	Position forwards half a page.
-H	Position backwards half a page.
<u>T</u> OP	Position to top of rule.
- -	
<u>B</u> OTTOM	Position to bottom of rule.
++	
+ <i>nnnn</i>	Position forwards <i>nnnn</i> lines (maximum 4 digits).
- <i>nnnn</i>	Position backwards <i>nnnn</i> lines (maximum 4 digits).
<i>nnnn</i>	Position to line <i>nnnn</i> .
X	Position to the line marked with "X".
Y	Position to the line marked with "Y".
<u>S</u> PLIT[- + <i>nn</i> ++ - -]	Use of positioning commands in split screen.

In split screen mode, all positioning commands must be preceded by an "S" (for Split Screen). See further information in the section Split Screen Commands.

Line Commands

In addition to the editor commands, the following line commands can be used when editing a processing rule:

Command	Function
.C(<i>nnnn</i>)	Copies the line in which the command was entered.
.CX(<i>nnnn</i>) .CY(<i>nnnn</i>)	Copies the X-marked or the Y-marked line. See also the commands ".X" and ".Y" in the following section.
.CX-Y(<i>nnnn</i>)	Copies the block of lines delimited by the X and Y markers. If the direction indicator is "+", the copied lines are placed after the line in which the command was entered. If the direction indicator is "-", the copied lines are placed before the line in which the command was entered.
.D(<i>nnnn</i>)	Delete line or lines. The default is 1 line.
.I(<i>n</i>)	Inserts <i>n</i> empty lines. With the next ENTER, lines that are left blank are eliminated again.
.I(<i>obj,ssss,nnnn</i>)	Inserts an object contained in the current library or in the steplib into the source. The " <i>ssss</i> " entry can be used to indicate the line number at which the insert operation is to begin. The " <i>nnnn</i> " entry can be used to indicate the number of lines to be inserted. See the more detailed description of the .I line commands in the section Line Commands.
.J	Joins the current line with the next line. If the resulting line length is greater than the length of the editor screen line, the line is marked with "L" and must then be separated again using the ".S" command (see below), before it can be modified.
.L	Undoes all modifications that have been made to the line since the last time ENTER was pressed.
.MX .MY	Moves the X-marked or the Y-marked line. See also the commands ".X" and ".Y" below.
.MX-Y	Moves the block of lines delimited by the X and Y markers. If the direction indicator is set to "+", the moved lines are placed after the line in which the command was entered. If the direction indicator is set to "-", the moved lines are placed before the line in which the command was entered.
.P	Positions the line marked by this command to the top of the screen.
.S	Splits the line at the position marked by the cursor.
.W	Inserts <i>n</i> empty lines. With the next ENTER, lines that are left blank are eliminated again.
.X	Marks a line, or the beginning of a block of lines, to be processed.
.Y	Marks a line, or the end of a block of lines, to be processed. Note: If both the commands ".X" and ".Y" are applied to one line, it is treated as being marked with "X" and with "Y"; the line marker actually shown to reflect this status is a "Z".

